ALTERNATIVE GROUP TRIP PLANNING QUERIES IN SPATIAL DATABASES

SHAHUL SHAIK
Bachelor of Technology, GITAM University, 2017

A thesis submitted
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

ALTERNATIVE GROUP TRIP PLANNING QUERIES IN SPATIAL DATABASES

SHAHUL SHAIK

Date of Defence: May 15, 2020

Dr. W. Osborn Thesis Supervisor	Associate Professor	Ph.D.
Dr. J. Rice Thesis Examination Committee Member	Professor	Ph.D.
Dr. Y. Chali Thesis Examination Committee Member	Professor	Ph.D.
Dr. H. Cheng Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

Dedicated to my beloved parents for their endless support, encouragement and sacrifices.

Abstract

Trip Planning Queries are considered as one of the popular services offered by Location-Based Services. We propose a new query type called an Alternative Group Trip Planning Query (AGTPQ) which is an extended version of Sequenced Group Trip Planning Queries (SGTPQs). Given a set of users' source locations and destination locations and a sequence of Categories of Interest (COIs) that the users want to visit, an AGTPQ generates a new COI sequence order using one of the proposed techniques and finds an optimal trip starting from the source locations, passing through the new sequenced COI order and ending at the destination locations. We propose three approaches: Permutation Strategy on Sequenced Group Trip Planning Queries (PSGTPQs), Greedy Strategy on Sequenced Group Trip Planning Queries (GSGTPQs) and Random Strategy on Sequenced Group Trip Planning Queries (RSGTPQs). We compare the results of our proposed strategies with the PGNE strategy through experimental evaluation.

Acknowledgments

I would like to convey my sincere gratitude to my supervisor Dr. Wendy Osborn for her continuous support of my master's degree, for her guidance, motivation, patience and immense knowledge.

I also want to thank my committee members Dr. Jackie Rice and Dr. Yllias Chali for their constant support and motivation. I would like to thank Dr. Howard Cheng for being the Examination chair of my thesis defense.

I would like to express my gratitude to the School of Graduate Studies (SGS) of the University of Lethbridge for their financial support. It is a great pleasure to thank my parents and my sister for believing and encouraging me throughout my life.

I would also thank my friends Ajay and Akalanka for making my life more easier.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

The proliferation of Geographic Information System (GIS) has lead to an increased interest in different types of Spatial Database Management Systems (SDBMSs) [26]. For example, Geographic Information Systems (GIS) are spatial database management systems that visualize and analyze geospatial data, which is spatial data about the objects that are positioned in and on the earth's surface [21]. Spatial data describes the size, location and shape of objects like restaurants, gas stations, hospitals and police stations on the earth. In general, simple types of spatial data include the point, line string, circular string, compound curve, polygon, and curve polygon. Composite types of spatial data include multipoint, multiline string, multi-polygon and geometry collection [26].

A spatial query is a unique database query which is supported by Spatial Databases. Two types of spatial queries include the Range Query and the Nearest Neighbour (NN) Query. Spatial relationships play an important role in spatial query processing by examining spatial objects with respect to each other (e.g. find all the restaurants within 1km from a particular retail store). In addition, spatial indexing which indexes spatial data by location in multi-dimensional space, is an efficient approach to retrieve data from the spatial database [26].

A spatial network is another type of spatial data [26]. It is represented in the form of a graph which consists of nodes (i.e. data points) and edges, that are positioned in space. Spatial networks are applied to various fields like power grids, transportation systems (i.e.

road networks), social networks and mobile networks. The primary idea of the spatial network is to establish connectivity between nodes, where objects and query movement is restricted to a pre-defined route. The shortest network path is calculated based on the connectivity of the network rather than the positions of the objects themselves.

Location-Based Services (LBS) have become more prevalent due to the proliferation of position technology and wireless communication [14]. Location-based services, also called location-dependent information services, provide services to users based upon their current location and are widely used in applications like mobile and wireless computing services. Spatial queries like Range Queries and Nearest Neighbour (NN) Queries are also processed by Location-based services by obtaining information on the locations of the queries (i.e. users) and objects.

In the rest of the chapter, we begin by discussing the motivation behind our research area in Section 1.1. Section 1.2 presents the contributions of this thesis to the field of study. Section 1.3 discusses the organization of the remaining chapters of this thesis.

1.2 Motivation

Initially, researchers only focused on Range Queries, Nearest Neighbour Queries and their variants [15]. The increase in demand for new types of queries to support modern spatial database systems and applications lead to the development of Trip Planning Queries (TPQs) [15]. In a Trip Planning Query, a user specifies a source location S, a destination D and a defined sequence C of Categories of Interest (COIs) (e.g. banks, libraries and parks) that the user wants to travel. A TPQ processing strategy retrieves an optimal route starting from S, passing through one Point of Interest (POI) from each Category of Interest in C and ending at the destination D.

TPQs are recognized as an essential part of LBS [2]. TPQs are also considered as a generalization of the Traveling Salesman Problem (TSP) which is an NP-hard problem [15]. Subsequently, several types of TPQs were proposed like Group Trip Planning Queries

(GTPQs) [12], Sequenced Group Trip Planning Queries (SGTPQs) [2] and Dynamic Group Trip Planning Queries (DGTPQs) [28].

In these new scenarios, we now have a group of users with their set of starting locations S, their set of destination locations D and a sequence of Category of Interests (COIs) that the users want to travel through together. A Sequenced Group Trip Planning Query (SGTPQ) strategy (such as PGNE [2] summarized in section 3.4) will find an optimal path starting from the each user's source location in S, passing through only one Point of Interest (POI) from the sequenced COI set and ending at their destinations in D. There are only a few proposed approaches that have studied the SGTPQ problem, and those existing approaches have one or more of the following limitations:

- Existing approaches randomly consider spatial networks where POI information is only located at the nodes, and not along the edges. This is not realistic in the real world and may lead to incorrect trip distances being generated, since the POIs being chosen are not in the right locations.

- Existing approaches only use a subset of COIs, and not all of the COI information contained in the spatial network. This is not realistic since many COIs could be chosen for a trip.

- Previous approaches are very costly with respect to computation to process the SGTPQs.

- Some existing approaches are only considered in Euclidean space and do not take the spatial network into consideration.

- Existing approaches do not consider other COI orderings for generating trips that may be more optimal than the trip originally submitted.

Therefore, we propose a new form of a Sequenced Group Trip Planning Query called the Alternative Group Trip Planning Query (AGTPQ) which generates a new and ideally

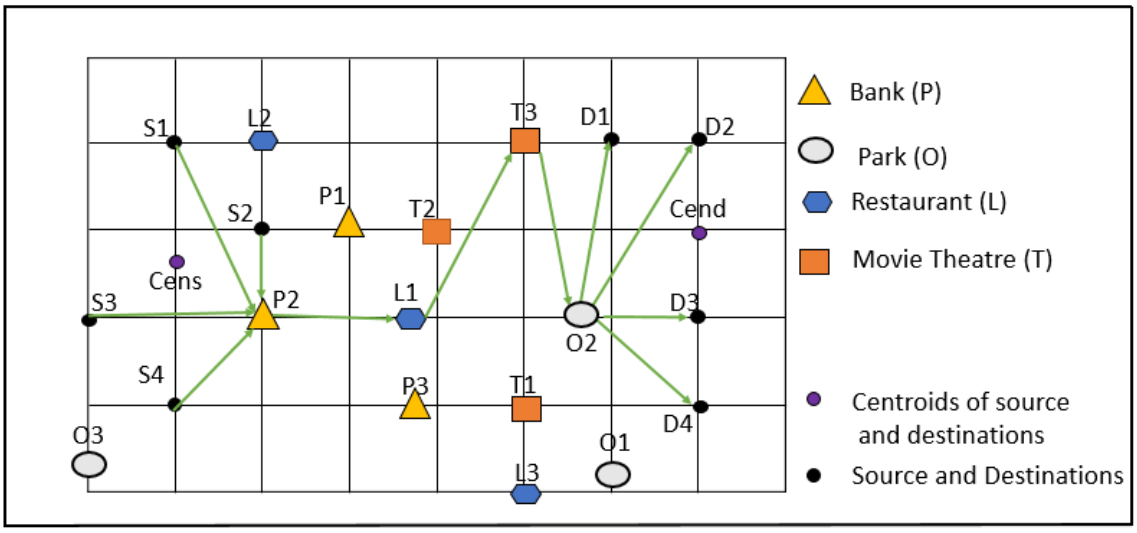

Figure 1.1: Example of AGTPQs

shorter trip by considering other COI orderings. We also propose three techniques with a goal to generate a more optimal (i.e. shorter) trip. Those strategies are Permutation Strategy on Sequenced Group Trip Planning Queries (PGTPQs), Greedy Strategy on Sequenced Group Trip Planning Queries (GGTPQs) and Random Strategy on Sequenced Group Trip Planning Queries (RGTPQs). From the results of our experiments, we observe that our AGTPQ strategies produce a 20% shorter trip than the trip found with the original COI sequence by the PGNE strategy.

Figure 1.1 illustrates the problem scenario with an example. Let us assume a group of four friends are staying at four different locations (e.g. different offices) S={S1, S2, S3, S4} and they are interested in travelling together to: first a Restaurant, second a Park, third to a Bank and finally to a Movie before they travel to their different destination locations D={D1, D2, D3, D4}. The type of places to visit like the Bank, Restaurant, Movie Theatre and Park are called Categories of Interest (COI). A Point of Interest (POI) is a specific instance of a COI. In Figure 1.1, POIs P1, P2, P3 belong to the Bank COI, POIs O1, O2, O3 belong to the Park, POIs L1, L2, L3 belong to the Restaurant COI and POIs T1, T2, T3 belong to the Movie Theatre COI. Our AGTPQ strategies will retrieve a new visiting order that ideally produces a shorter trip. For example, the new visiting order is to visit first a

Bank, second a Restaurant, third a Movie and finally a Park. The corresponding sequence of POIs (P2, L1, T3, O2) produces a shorter trip than the original one obtained by a SGPTQ processing strategy like PGNE.

1.3 Contributions

Our work contributes to the Sequenced Group Trip Planning Queries (SGTPQs) research field as follows:

- We present a new form of query called the Alternative Group Trip Planning Query (AQTPQ) which is based on the SGTPQ.
- The existing California dataset does not contain POI co-ordinate information. Therefore, we present an updated California data set. The new California dataset contains co-ordinate information for each POI lying on the appropriate edge, for all POIs in the dataset, not just for those that belong to a subset of COIs.
- We propose three strategies for answering the AGTPQs with the new data, which are the Permutation Strategy, Greedy Stratergy and Random Strategy.
- We conducted a set of experiments on our proposed query and strategies using the modified real-world California dataset to determine the effectiveness and efficiency of our proposed strategies.

1.4 Organization

The structure of the rest of this thesis is as follows.

In Chapter 2, we present a background study of the existing strategies for processing the various Nearest Neighbour, Trip Planning, and Sequenced Trip Planning Queries for Location Based Services.

In Chapter 3, we present an overview of the Multiple Query Method for processing a Group Nearest Neighbour and the PGNE approach for SGTPQs. Then, we present our

proposed AGTP query and corresponding strategies in detail for processing the AGTPQs for LBS.

In Chapter 4, we describe the parameters that we used in our experiments and the metrics to evaluate the performance of our strategies. Then, we present the evaluation results for the three proposed approaches versus the PGNE approach.

Finally, in Chapter 5, we present the conclusion of the thesis and identify future works of our research work.

Chapter 2

Background Study of Related Work

In this chapter, we present the summarized background study of our research work. This chapter is organized as follows: Section 2.1 presents existing work of Nearest Neighbour (NN and kNN) strategies, while Section 2.2 presents existing work of Aggregate Nearest Neighbour (ANN) queries. Finally, Section 2.3 presents the existing work of Sequential Group Trip Planning (SGTP) queries.

2.1 Nearest Neighbour Query Strategies

The Nearest Neighbour (NN) search is a proximity search that retrieves the closest point or object to the query point in a point or object set. The K-Nearest Neighbour search (k-NN) retrieves data objects which are closer to the query point. Here k is defined as the number of NNs. Most of the previous works used the Euclidean distance measure to determine the closest object(s).

Roussopoulos et al. [21] studied the problem of Nearest Neighbour queries in Geographic Information Systems. The authors proposed an efficient branch-and-bound R-tree traversal algorithm to find the K-Nearest Neighbours.

The authors proposed an algorithm to find the first NN, and then proposed another generalized algorithm to find k-NN. The algorithm starts processing from the R-tree [10] root node traversing down the tree using the depth-first traversal method. While traversing to the descendent nodes, the algorithm uses two different metrics called MINDIST and MINMAXDIST to sort the Minimum Bounding Rectangles (MBR) that are encountered, and

to prune to MBRs which contains objects that cannot be part of the k-NN result. Minimum Bounding Rectangle is the smallest rectangle that contains a group of objects. The MINDIST metric is based on the minimum distance from the query point to the objects contained in the MBR. The MINMAXDIST is the minimum value of all the maximum distances between the objects and the query point. The authors' MINMAXDIST metric avoids visiting the unnecessary MBRs by setting an upper bound on distance.

The authors conducted a set of experiments on both distance metrics and the proposed algorithm. The results show that the MINDIST metric is more effective than the MINMAXDIST metric in terms of accessing the number of points.

Papadias et al. [19] studied the problem of query processing in Spatial Network Databases (SNDBs), including NN and k-NN queries. According to the authors, the existing approaches only focus on Euclidean space for processing the spatial queries. So, they developed algorithms that combine Network and Euclidean information to overcome the existing limitation.

The authors developed two algorithms called Incremental Euclidean Restriction (IER), and Incremental Network Expansion (INE). The IER algorithm uses an incremental algorithm to retrieve the k-NNs for the query. It first computes the first NN from the query point using the Euclidean distance and then the Network distance is calculated from the same query point to its first NN. Using the Network distance, a circle bound is formed with the query as its centre and the length of its radius equal to the calculated Network distance. Then, the points inside the circle are processed incrementally and the remaining points outside the circle are pruned. For the points that are lying inside the circle, their network distances are computed and compared against the other points. Then, all the kNNs are sorted based on their network distances to the query.

One limitation of the IER algorithm is that it processes redundant network computations. So, the authors proposed the INE approach to overcome the IER algorithm limitation by searching the nearest nodes. The INE algorithm first searches the query edge and locate

any points lying on that edge. The query edge is the edge where the query lies on it. If no points are lying on the query edge, then the INE algorithm searches the nearest node to find the points. The first NN distance to the query acts as a bound to limit the search space. Then, the INE algorithm searches the other nearest nodes incrementally inside the search space to find any points that can generate a lower distance than the first NN.

The authors conducted an experimental evaluation and comparison of the proposed algorithms. The results indicate that the INE approach is better than the IER approach in terms of reducing the I/O computations due to redundant network computations. The IER approach is found to perform in denser networks.

Kolahdouzan and Shahabi [13] also studied the problem of k-NN search. According to them, the existing approaches are computationally expensive for computing the distance between objects. The authors proposed a method that uses the Voronoi diagram to overcome this limitation. A Voronoi diagram [13] is a partition of a region into small regions close to each of a given set of objects.

The authors' approach divides a large spatial network into small Voronoi regions and then the network distance is precomputated within each regions. The authors state that the pre-computation within the regions will save on storage and computation costs. Also, the authors proposed new properties of network Voronoi polygons (NVP) to prove that the k-NNs of the query object are within the adjacent NVPs of the previously explored nearest neighbours. The experimental results show that the authors' approach outperforms the existing approaches in terms of response time and generating low computation and space complexities.

Du and Ji [33] proposed a road network model for processing k-NNs. The proposed approach integrates mobile agent concepts with the INE algorithm [19]. The mobile agents are controlled by a Central Control Center which is in charge of the whole operation model. The authors' network is divided into several local areas. The strategy first sends the user queries to the Central Control Center by using mobile agents. Then the Central Contol Cen-

ter sends a local area agent to the corresponding local area. The local area agent executes the query in the corresponding local area with the existing INE algorithm. The results are sent to the Central Control Center and then to users via other mobile agents.

2.1.1 Continuous Nearest Neighbour Search

The continuous spatial query is utilized in Location-Based Services, which will continuously find a result for a moving query and need continuous updates [35, 30]. For example, a taxi driver may want to locate the customers' position(s) as they move frequently, which needs continuous updates for the query answer.

Tao et al. [30] study the problem of a Continuous k-Nearest Neighbour (CKNN) search over a query line segment.

According to the authors, existing methods have the following limitations:

- Some strategies use a heap to store intermediate results, which will lead to buffer thrashing if the heap becomes too large.
- Some strategies use a sampling method to obtain the outcome, which may lead to missing some NNs.
- A few strategies only work well for smaller queries and datasets due to CPU overhead.

The authors proposed an efficient solution to overcome the above problems of continuous nearest neighbour query processing over a line segment. First, the algorithm finds all split points. Each split point is formed by the intersection between the query line segment and the perpendicular bisector (i.e., a line between two current nearest neighbour points). Then, the nearest neighbour to the split point is identified. If any new data is updated in the point set, and if the new point covers a split point, then it becomes the nearest neighbour to the split point. If the point does not cover any of the split points, it cannot become the nearest neighbour. The authors' proposed approach uses an R-tree [10] to enhance the

performance by pruning the search space. The R-tree is traversed in the following manner. For any intermediate node, only the subtrees are visited that contain any qualifying data point. According to the authors, the proposed CNN query algorithms can be used for both CKNN and trajectory nearest neighbour search. The CKNN query retrieves the k-NNs for every split point on a query line segment. The trajectory NN applies the process to several line segments and retrieves the NN for each split point on every segment.

The authors compare their strategy against the TP method [29] with both uniform and real data sets. Experimental results show that when the query length increases then the number of node accesses and CPU time also increases and as a result the performance of CNN degrades.

Veeresha et al. [32] proposed a Hybrid Spatial Air Index (HSAI) for processing continuous k-NN queries in road networks to enhance query performance. Existing approaches do not handle a large number of points, links and cache management errors in the broadcast system. The proposed Hybrid Spatial Air Index (HSAI) use both XOR-based network coding and Adaptive Cooperative Caching (ACC) benefits.

The authors divide the road network into grid cells by using a grid partition technique. On the server-side, HSAI objects are integrated and broadcast-encoded into a Hybrid broadcast schedule, and combined with XOR-based network coding into the broadcast scheduling program. Adaptive Cooperative Caching (ACC) predicts the movements of the client and makes cache replacement decisions on the objects.

In the proposed method, when a user issues a query from their mobile device, the cache is consulted first. If the query result is not found then the query request is sent to the region query directory (QD). If the query result is not found here then the user tunes in to the channel to retrieves the result. The authors executed Dijkstra's algorithm [6] based on the query result and process the shortest path between the query and nearest neighbour.

The authors compared their HSAI algorithm with an existing approach. Results show that the HSAI approach gives better performance with respect to the tuning time, energy

efficiency and access latency.

Kyriakos et al. [18] study the problem of continuous monitoring of nearest neighbours in highly dynamic environments, where both objects and queries move arbitrarily and frequently. According to the authors, existing methods have one or more of the following limitations:

- Previous methods only deal with snapshot queries over static data.
- Existing methods assume the use of the Euclidean distance metric.
- Previous approaches provides only approximate CKNN queries.
- Existing approaches cannot handle updates due to changes in the edge weights.
- Existing techniques are inapplicable to road networks.

The authors proposed an efficient solution to overcome this problem with an incremental monitoring algorithm and a group monitoring algorithm. When a query arrives at the system, the incremental monitoring algorithm retrieves an initial NN and expands the network around the query until all k-NNs are found. The initial query result is computed as follows. First, the query is initialized as a root of an expansion tree. Second, NNs are inserted on the query edges according to their edge weights. Third, verified nodes (i.e. when the distance between the node and the query is known) are removed from the heap. If a new unverified adjacent node can be reachable through a verified node then its key distance (i.e. sum of the distance between the node and query plus the adjacent node weight) is updated and inserted into the heap. Otherwise, if an unverified node is already in the heap then an adjacent node is made as a child of a verified node in the query tree. Then unverified nodes are removed from the heap and the shortest distance from the query to each node is stored back into the heap in the form of the expansion tree.

The authors' group monitoring algorithm is based on a shared execution paradigm and integration with the incremental monitoring algorithm. It groups the queries that reside

along the same path between two consecutive intersections. The group monitoring algorithm monitors the k-NNs of the intersections to produce efficient results of all the queries along the path.

The authors provide an experimental evaluation of their algorithms. The group monitoring algorithm outperforms existing methods in terms of running time and CPU time. The group monitoring algorithm CPU time does not change based on object speed, but the incremental monitoring algorithm running time changes slightly. The incremental monitoring algorithm consumes more space than the group monitoring algorithm because it stores the tree of every query.

Hicham et al. [9] proposed algorithms to process a Continuous Aggregate Nearest Neighbour (CANN) query for moving objects in the spatiotemporal data systems. The result of a CANN query is a set of k-nearest objects which have the shortest distance to the set of fixed points or landmarks. The proposed method improves the group nearest neighbour query (GNN) approach, which handles only snapshot queries.

The authors proposed two efficient algorithms for processing CANN queries. The Holistic CANN (H-CANN) algorithm calculates the aggregate distance from the objects to all landmarks and decides whether to remove any objects from being part of the query result. The Progressive CANN algorithm (P-CANN) algorithm maintains and calculates the result of the CANN method for a certain time. P-CANN sets the threshold for every landmark and decides whether to remove any moving objects from the answer. A few policies are introduced in P-CANN to define the order of landmarks for calculating the aggregate distance.

The experimental results of the proposed algorithms show that P-CANN outperforms H-CANN and the conceptional partition algorithm [17] in terms of low memory overhead.

2.2 Aggregate Nearest Neighbour Queries

Aggregate Nearest Neighbour (ANN) search is a generalized form of the Nearest Neighbour search, and is also known as Group Nearest Neighbour (GNN) search. The ANN query depends on the multiple query points rather than a single query point which is used in existing work. Given a set of query users, Q=$\{q_1, q_2\}$, and data points, P=$\{p_1, p_2, p_3\}$, the ANN query [8] retrieves the data point from P that minimizes the total travel distance to all query points in Q.

Papadias et al. [8] proposed GNN query processing strategies that aim to reduce the sum of the Euclidean distances from the users' locations to a Point of Interest (POI). The following three proposed algorithms process a GNN query by assuming the query points can fit into the memory and can be indexed by an R-tree [10].

- Multiple Query Method (MQM),
- Single Point Method (SPM),
- Minimum Bounding Method (MBM).

The Multiple Query Method combines the results of each incremental nearest neighbour search for every point in Q. It employs a threshold algorithm to find a better optimal point. The Single Point Method uses a single traversal method to process the GNN queries and to avoid the multiple accesses of the Multiple Query Method. The Minimum Bounding Method performs a single query and it uses the R-tree [10] to reduce the search space of Q.

The authors also proposed disk-resident methods to handle queries with large results that can not fit into memory and handles both non-indexed and indexed query points.

- Group Closest Pairs Method (GCP),
- File-Multiple Query method (F-MQM),
- File-Multiple Bounding method (F-MBM).

The Group Closest Pairs Method assumes that Q is indexed by an R-tree and uses pruning rules to process the GNN queries. The proposed F-MQM and F-MBM methods do not require any indexing of Q. The F-MQM methods split the query Q into small sets. For each set, the GNN is performed, and the final results are compared using MQM. The proposed F-MBM method is an extension of SPM and MBM where Q does not fit into memory.

The authors conducted a set of experiments on their proposed methods. From the results, the authors observe that the GCP method is the worst performing method in all cases and the F-MBM method is the best performing method among all the proposed algorithms only when the data is partitioned into a small set of groups.

Yiu et al. [34] studied the problem of ANN queries in road networks. The authors state that the existing work on ANN queries only apply to Euclidean distance. Therefore the authors proposed three approaches for processing ANN queries to overcome this limitation:

- Incremental Euclidean Restriction (IER),
- Threshold Algorithm (TA), and
- Concurrent Expansion (CE).

In all algorithms the data points are indexed by the R-tree [10]. The IER algorithm iteratively retrieves each nearest point and computes the Euclidean aggregate distance of each NN. This process terminates when the result can not be improved. The TA approach incrementally explores the network and terminates the search using the top-k aggregate query processing technique. The CE algorithm works similarly to TA but it avoids shortest path computations.

The authors conducted an experimental study to evaluate the efficiency of their methods in a real road network by using the shortest path distances between network nodes. The results show that the TA outperforms IER and CE in terms of different aggregate functions, IER outperforms CE and TA in terms of page access and response time and CE is the worst performance in terms of different aggregate functions, page access and response time.

Sultana et al. [27] studied the problem of GNN queries in the presence of obstacles. The authors proposed a generalized form of GNN queries called obstructed group nearest neighbour (OGNN) queries. The existing approaches only focused on the Euclidean and road network spaces by ignoring the impact of obstacles such as parks and buildings when processing the aggregate distance. To overcome the limitations of the existing research, the authors proposed a Multi-Point Aggregate Obstructed Distance (MPAOD) approach to process OGNN queries.

The authors computed the aggregate obstructed distances using a visibility graph. The visibility graph contains nodes and their associated POIs, obstacle vertices, and query points. An edge has no obstacles if there is clear visibility between two nodes on that edge. If there are any obstacles then that edge is not taken into account for the aggregate group distance computation. The authors also introduced geometric pruning techniques to reduce the search space and computational overhead.

The authors conducted an experimental evaluation using both real and synthetic data sets against the proposed approach to demonstrate its efficiency in terms of CPU time and IO access.

Safar [22] studied the problem of Group k-Nearest Neighbour (GKNN) queries in spatial network databases. The authors state that existing approaches incur a high computational cost by accessing the same node(s) and point(s) multiple times. The author proposed a new approach to overcome the limitations. The approach is based upon the network distances needed to retrieve the GKNN query. Also, the authors' approach uses the Voronoi diagram together with a progressive incremental network expansion method for the determination of the inner network distances.

The authors conducted experiments and noted that the GKNN query execution time depends on the size of query objects and the density of the POIs.

2.3 Trip Planning Queries

A Trip Planning Query (TPQ) is a more recent form of query proposed in Spatial Databases, in particular for use in location-based services. Given a source location S, destination location D and set of Categories of Interest (COI), a Trip Planning Query [15] will retrieve an optimal path starting from the user's source location S, travelling through a sequence of Points of Interest (POI), where each POI is from a corresponding COI, to the user's destination D. The work in TPQs helped to motivate research in new forms of queries like Group Trip Planning Queries (GTPQs) and Sequenced Group Trip Planning Queries (SGTPQs).

Section 2.3.1 summarizes the related research work in Trip Planning Queries, while Section 2.3.2 summarizes the related work in Group Trip Planning Queries and Section 2.3.3 summarizes the related work in Sequenced Group Trip Planning Queries.

2.3.1 Trip Planning Queries

Li et al. [15] were the first to introduce the Trip Planning Query (TPQ). The authors proposed two approaches called the Nearest Neighbour Algorithm (A_{NN}) and the Minimum Distance Algorithm (A_{MD}) to process the Trip Planning Query.

The Nearest Neighbour Algorithm (A_{NN}) forms a trip by iteratively visiting to the NN of the previous vertex that was added to the trip. The Minimum Distance Algorithm (A_{MD}) chooses a set of vertices from each COI in such a way that the cost of each vertex is minimum among all vertices belonging to the respective category. Then the trip is generated by travelling through the selected vertices in the NN order.

The authors also discussed the implementation of TPQs in both Euclidean space and road networks. The authors explained about the adaption of A_{NN} and A_{MD} algorithms in the Euclidean space where the POIs and nodes are indexed using the R-tree. Also, the authors explained the adaption of A_{NN} and A_{MD} algorithms in the road network by using the extended version of the Dijkstra's algorithm [6].

The authors conducted experiments using both real-world datasets and synthetic datasets. The results show that A_{MD} outperforms than the A_{NN} in terms of generating less trip distance.

Sharifzadeh et al. [24] study the problem of Optimal Sequenced Route (OSR) query in vector spaces. According to the authors, Dijkstra's algorithm is impractical for the real world due to the large network graph. So, the authors proposed approaches to overcome this limitation. The first approach is called the Light Threshold-based Iterative algorithm (LORD) which employs different threshold values to prune POIs that cannot be in the optimal route. The second proposed approach is an extension of LORD which uses an R-tree [10] to refine the POIs that cannot be part of the optimal path and then builds an optimal route from the end point to the starting point. Both proposed methods are not applicable in the metric space, so the authors proposed another approach called the PNE approach that processes NN queries on various point types to form an optimal route.

The authors conducted several experiments on the proposed approaches against existing approaches. Form the results, the authors state that the proposed approach outperforms the existing approaches in terms of required workspace and processing time.

Sharifzadeh et al. [25] studied the problem of the Optimal Sequenced Route (OSR) query using Voronoi Diagrams in both vector and metric space. The existing approaches only focused on vector space. The Voronoi-based Optimal Sequenced Route query (OSRV) approach uses pre-computed Additively Weighted Voronoi (AW-V) diagrams to answer an OSR query.

The authors conducted experiments using real-world datasets. From the experiments, the authors found that the proposed approaches outperform the existing approaches in terms of response time.

Chen et al. [5] studied the problem of trip planning search in Geographic Information Systems (GIS). The existing approaches process TPQs by only allowing a user to visit in sequence. The authors proposed the multi-rule partial sequenced route query to overcome

this limitation of existing methods. The authors proposed two heuristics. In their first heuristic, the authors proposed a nearest neighbour-based partial sequenced route (NNPSR) query algorithm which uses a topological sort to combine multiple travel rules and executes the nearest neighbour search for the complete trip.

The secondly proposed heuristic reduces the total trip distance by integrating NNPSR with the Light Optimal Route Discovery (LORD) approach [24]. The third heuristic generates efficient trip plans by proposing an advanced A^* algorithm. Also, it considers the destination of the user in terms of generating a shorter trip.

The authors conducted experiments to evaluate the performance of their proposed approaches. From the results, the author state that the proposed approaches outperform the LORD existing approach in terms of generating a shorter trip and achieving lower processing time.

Costa et al. [7] studied the problem of processing optimal time-dependent route queries in a road network. The existing approaches do not consider the time factor, and only focus on finding an optimal route. The authors proposed an approach which consider the amount of time spent by the user at each POI. They also use entry and exit nodes and A^* algorithm to find the shortest path and to reduce the search space. The authors evaluated their proposed approach against an existing approach, and found that the proposed approach outperforms the existing approach in generating faster trips.

Lu et al. [16] study the Multi Request Route Planning (MRRP) problem in urban environments. According to the authors, in route planning queries, the existing approaches mainly focus on the geographic properties of the POIs, but not about the POIs that satisfy the user needs. Therefore the authors' goal is to produce an optimal route for a user where the actual number of stops is less than the number of actual COIs. Their example is: a user wants to travel to the nearest bank to withdraw money and then travel to a convenience store. Since most convenience stores have an ATM, the user does not have to go to the bank - they can just go to the nearest convenience store.

The authors' proposed framework consists of two modules: the Planning module and the Refinement module. The Planning module consists of four proposed approaches:

- k-Nearest Neighbour with Most Services (kNNMS)
- k-Minimum Distance with Most Services (kMDMS)
- Embed-Based Approach (EMB)
- k-Request A^* with Most Services (kRAMS)

The Refinement module consists of two approaches:

- Route Rearrangement
- Route Replacement

The Planning module approaches are used for planning a preliminary route. All approaches use pruning and caching strategies. The Refinement module approaches improve on the preliminary route.

The authors evaluated the performance of their framework by using the Oldenburg, Tainan and NewYork datasets. The experimental results show that the kRAMS approach significantly improved the planning of the route and efficiency in terms of total distance, number of POIs visited and processing time than the other proposed approaches.

2.3.2 Group Trip Planning Queries

A Group Trip Planning Query [11] is a recent extension to trip planning queries. A Group Trip Planning Query is a trip that minimizes the total travel distance for several members in the group. For example, given a set of source locations S, destination locations D and set of Categories of Interest (COIs), a Group Trip Planning Query finds a path that minimizes the total travel distance for all members starting from their individual source locations in S, visiting through a set of COIs as a group and ending at their individual destinations in D.

Hashem et al. [11] were the first to introduce this new query type in both Euclidean space and road networks. According to them, the existing approaches process Trip Planning Query for only a single user.

The authors propose two approaches that use geometric ellipse properties to refine the search space of Points of Interest (POIs) and to reduce computations. The authors mention that the search space size is proportional to the I/O and computational overhead because the smaller space size has fewer POIs than a larger space.

Their first approach is based on a Three-Step Range-Based algorithm that retrieves POIs within the elliptical search space and also finds an optimal route based on the retrieved POIs. The approach first retrieves an initial set of POIs using the pruning rules and then refines this set using the refined search space. Then, the final or optimal set of POIs from the refined set are found using dynamic programming. The second approach uses an incremental strategy, only accessing the required POIs that are needed to find the optimal result.

The authors conducted a set of experiments with both real and synthetic datasets. From the results, the authors state that the proposed techniques outperform the existing approaches in terms of I/O and computational overhead.

Rayhan et al. [20] study an improvement to their previously proposed GTP [11] strategies. According to them, the GTP queries experience are computational overhead because of unnecessary group trip distance calculations.

First, the authors calculated the group total travel distance by computing each user's distance in the group independently, but this process incurs high query processing time because of the retrieval of the same points multiple times. Therefore the authors proposed two approaches: Iterative Approach (IA) and Hierarchical Approach (HA).

The Iterative Approach uses a group nearest neighbour (GNN) query to retrieve the set of points which has the smallest total distances for the group. IA first retrieves all first GNNs of all COIs with respect to the users' source locations until the last sequenced COI. The GNN for the last COI is determined with respect to the group members' destination

locations.

Then with this information, IA calculates the initial total trip distance using the GNN found so far. Afterwards, it backtracks to find any other trips (containing other GNNs) which are shorter than the initial trip. However, the authors state that the IA method accesses the same POIs multiple times when finding other trips shorter than the initial trip. Unlike the IA method, the HA method uses a single database traversal to reduce the multiple retrievals of the same POIs.

For the experiments, the authors evaluated the performance of their approaches using both real and synthetic datasets. The authors found that HA outperforms the IA in terms of incurring less processing time.

2.3.3 Sequenced Group Trip Planning Queries

The Group Trip Planning Queries (GTPQs) do not consider the visiting COIs in sequence - rather they only focus on minimizing the group's aggregate distance. The Sequenced Group Trip Planning Queries (SGTPQs) consider the COI visiting order along with focusing on minimizing the group's aggregate distance.

Ahmadi and Nascimento [2] were the first to study the processing of Sequenced Group Trip Planning Queries in road networks. According to the authors, existing approaches only focus on the generating group's minimum distance but not on visiting sequence. The existing Iterative Approach [20] are unable to produce optimal answers and yields high computational overhead. The authors proposed efficient approaches to overcome the existing limitation.

The first approach is a correction to the existing Iterative Approach [20] called Revised Iterative Approach (RIA) which uses the Single Point Method [34] to process GNNs. It reduces computational overhead by considering the location of the previously retrieved POI and group size to retrieve other points of interest for consideration.

The author's main approach in this paper is the Progressive Group Neighbour Expansion

(PGNE) approach, which also considers destinations of the users while retrieving POIs. PGNE reduces the search space based on the mixed Breadth-Depth First Search technique. It creates a set of partial group trips (PGTs) where each contains a potential portion of a trip from the users' source locations to a COI that is part way through the COI sequence. To find an optimal trip using a PGT, each PGT is extracted from the top of a priority queue and processed through replacement and progressive operations. In the replacement operation, the POI of the last COI in the PGT will be replaced with another POI, which is the next GNN from the same COI. The progressive operation will append the first GNN POI of the next remaining COI defined for the trip to the extracted PGT. The authors also proposed pruning strategies to prune the PGTs that can not be part of the optimal answer.

The authors evaluated their proposed strategies using both synthetic and real datasets. The authors evaluated PGNE versus RIA, with various group sizes, query areas, and number of COIs. The authors found that PGNE outperformed the RIA in terms of low response time and retrievals of fewer POIs.

Samrose et al. [23] study the problem of group optimal sequenced routes in road networks using location-based social networks. According to the authors, the existing approaches only focus on a single user trip planning query in Euclidean space.

The proposed approach retrieves an initial set of POIs for the required COIs using the users' source and destination locations. Then, elliptical pruning strategies are applied to the initial set of POIs. The authors calculate an ellipse for each group member from their locations and retrieve only the POIs lying inside the intersection of all ellipses. The authors also state that the POIs lying outside of the intersection of all the ellipses can not be part of the optimal answer.

The authors conducted experiments on their approach versus the naive approach. The results show that their approach outperforms the naive approach in terms of I/O and computation.

Tsatsanifos et al. [31] studied the problem of minimizing the total travel distance of

users by using a pre-defined mutual affinity factor. The proposed method calculates the shortest path between all nodes using Dijkstra's algorithm [6]. For example, a group of four friends A, B, C, D wants to travel together. If the user A wants to travel alone and B, C, D wants to travel together then the proposed method will retrieve a meeting point from the starting points of the users and finds the shortest path to the destination connecting the remaining users. In this proposed method, each user can specify with whom they want to travel to the destination along with connecting POIs. The authors proposed a new type of query called an Optimal Single Connecting Point (OSCP) query to process the above stated problem. The authors use an OSCP cost function and graph search approaches (e.g. branch-and-bound paradigm and Dijkstra algorithm [6]) to process the OSCP query.

The authors conducted experiments using real-world datasets. According to the authors, their approach scales better with respect to the number of travellers, query coverage area, self-affinity factor and answer size.

Barua et al. [4] proposed a new form of Sequenced Group Trip Planning Query called a Weighted Optimal Sequenced Group Trip Planning (WSGTP) query, where POIs are weighted based on popularity. The existing approaches only focus on retrieving the optimal path but does not focus on the POI popularity for processing the SQTPQs. A POI's popularity is based on previous user ratings and user reviews, which is extracted from the Trip Advisor and Google platforms. The authors defined that a POI which has a high popularity is called a high weighted POI. The authors' Weighted Optimal Sequenced Group Trip Planning Query processing strategy will retrieve the path from the source locations, through the sequenced COIs and then to the destination locations while both minimizing the group total travel distance and maximizing the weights (popularity) of the POIs.

Their strategy uses elliptical pruning strategies to limit the search area of POIs. It also uses a dynamic programming procedure to calculate the optimal path, and an R-tree[10] to index the POIs. The authors also proved that POIs outside the computed elliptical search area will not generate optimal results.

The authors carried out a set of experiments on the proposed approach. From the results, they show that their approach is efficient in reducing the computational time of the dynamic programming algorithm.

Ahmadi and Nascimento [3] also continue to study the problem of processing Sequenced Group Trip Planning Queries. The existing approaches process SGTPQs by accessing the same POIs multiple times which leads to the high computational cost. The authors proposed the Iterative Backward Search (IBS) approach to overcome the existing work limitation.

The IBS approach is based on a suffix optimality principle which is also proposed by the authors. The proposed suffix optimality principle makes re-use of the previously processed POIs. IBS first retrieves each all of the first nearest POIs based on the predefined sequence of COIs and then calculates the initial trip distance. Then it backtracks by finding the next nearest POIs of sequenced COIs to find other trips that are shorter than the current trip. In addition to considering the source and destination locations of the users, IBS also uses pruning techniques to reduce the query search space.

The authors conducted several experiments for the proposed approach against the existing techniques. From the results, the authors state that the proposed IBS approach outperforms the existing approaches with respect to the POI density, query area, group size, the number of COIs and the COI density.

Tabassum et al. [28] study the problem of Dynamic Group Trip Planning (DGTP) Queries in Spatial Databases. The existing approaches process GTP queries by assuming the group size is fixed or static. To overcome the limitation, the authors proposed a new type of GTP query called a Dynamic Group Trip Planning query where any group member can leave or join the group at any POI along the trip.

The authors proposed an approach called NAIVEDGTP, which uses dynamic programming for processing the DGTP queries in the Euclidean space. Their algorithm has two phases. The first phase retrieves all the POIs from the database for the initial trip. The

second phase has an update operation which updates the remainder of the trip according to the change in the group size in real-time. The authors also developed refinement techniques to reduce the search space of POIs which helps in reducing the computational overhead. Also, the authors used a cache to store the POI information instead of retrieving the same POIs multiple times from the database.

The authors conducted a set of experiments using both real and synthetic datasets. The NAIVEDGTP approach outperforms the naive approach by 99.74% times faster and requires less I/O time.

2.4 Summary

This chapter summarized the existing work of Nearest Neighbour search, Continous Nearest Neighbour search, Aggregate Nearest Neighbour search, Trip Planning Queries, Group Trip Planning Queries and Sequenced Group Trip Planning Queries. In the next chapter, we present our proposed strategies for Alternative Group Trip Planning Queries.

Chapter 3

Alternative Group Trip Planning Queries

In this section, three strategies are proposed for processing the Alternative Group Trip Planning Queries. The first strategy is called the Permutation Strategy. The second strategy is called the Greedy Strategy. The last strategy is called the Random Strategy. In the following sections, we discuss in more details the problem definition, sample network, generation of new data, methodologies we used and the proposed strategies.

The rest of this chapter is organized as follows. In Section 3.1, we describe the symbols that we used in our strategies and problem definition. In Section 3.2, we present a brief description of processing the Nearest Neighbour (NN) and Group Nearest Neighbour Strategies. In Section 3.3, we describe the sample network and present the generation of new data that we used in all three proposed strategies. In section 3.4, we present a brief description and running example of the PGNE approach that we used for processing the Alternative Group Trip Planning Queries. In Section 3.5, we describe our three proposed approaches: Permutation Strategy on Sequenced Group Trip Planning Queries (PSGTPQs), Greedy Strategy on Sequenced Group Trip Planning Queries (GSGTPQs) and Random Strategy on Sequenced Group Trip Planning Queries (RSGTPQs).

3.1 Preliminaries

In this section, we describe the terminologies, symbols and the definition of the problem that we are using in the remaining sections and chapters. We present the definitions for the

group total travel distance, Sequenced Group Trip Planning Query and Alternative Group Trip Planning Query below. Table 3.1 presents the symbols used in our definitions and algorithms.

Table 3.1: Definitions of symbols used in SGTPQs and AGTPQs.

n	Number of users
S= $\{S_1, S_2, S_3,..,S_n\}$	Starting locations of the users
D= $\{D_1, D_2, D_3,.., D_n\}$	Destination locations the users
m	Number of Category of interests
C= $\{C_1, C_2, C_3,...,C_m\}$	Sequenced of COIs to visit
P= $\{P_1, P_2, P_3, ., P_m\}$	Sequence of POIs for corresponding COIs
P_i	Instance of Ci
C_{ens}	Centroid of source locations of users
C_{end}	Centroid of destination locations of users
d_e	Euclidean distance between objects
d_n	Network distance between objects
MinDist	The current minimum distance calculated so far
k	Number of Nearest Neighbours

Definition 3.1. Given a set of n users, a set of source locations S = $\{S_1, S_2, S_3,..,S_n\}$ and destination locations D = $\{D_1, D_2, D_3,.., D_n\}$ and a sequence of m COIs C= $\{C_1, C_2, C_3,...,C_m\}$ that the users want to visit, the group total travel distance for visiting a trip sequence P= $\{P_1, P_2, P_3, ., P_m\}$ where Pi $\in$ Ci is defined as:

$$TD(P) = \sum_{i=1}^{n} d_n(S_i, P1) + n\sum_{i=1}^{m-1} d_n(P_i, P_{i+1}) + \sum_{i=1}^{n} d_n(P_m, D_i)$$

where $d_n(u, v)$ is the spatial network distance between locations u and v.

Definition 3.2. Given a set of n users, their starting locations S, a sequence of COIs C that they want to visit, and their destination locations D, the Sequenced Group Trip Planning Query will retrieve the Pi $\in$ Ci so that the total travel distance by all the users is minimal.

Definition 3.3. Given a set of n users, their starting locations S, a sequence of COIs C that they want to visit, and their destination locations D, the Alternative Group Trip Planning Query will retrieve the $Pi \in$ Ci from an alternative order of COIs. The alternative COI

sequence is determined by one of three proposed strategies (i.e Permutation, Greedy and Random strategies) so that the visiting order of total travel distance by all the users is minimal.

3.2 Nearest Neighbour and Group Nearest Neighbour Strategies

The Nearest Neighbour (NN) query is a query that will retrieve one or more data objects from the given set which are nearest to the query. Many proposed NN algorithms assume the use of the Euclidean distance metric to measure the distance between the objects and the query. However, other distance measures such as spatial network distance can be used depending on the application.

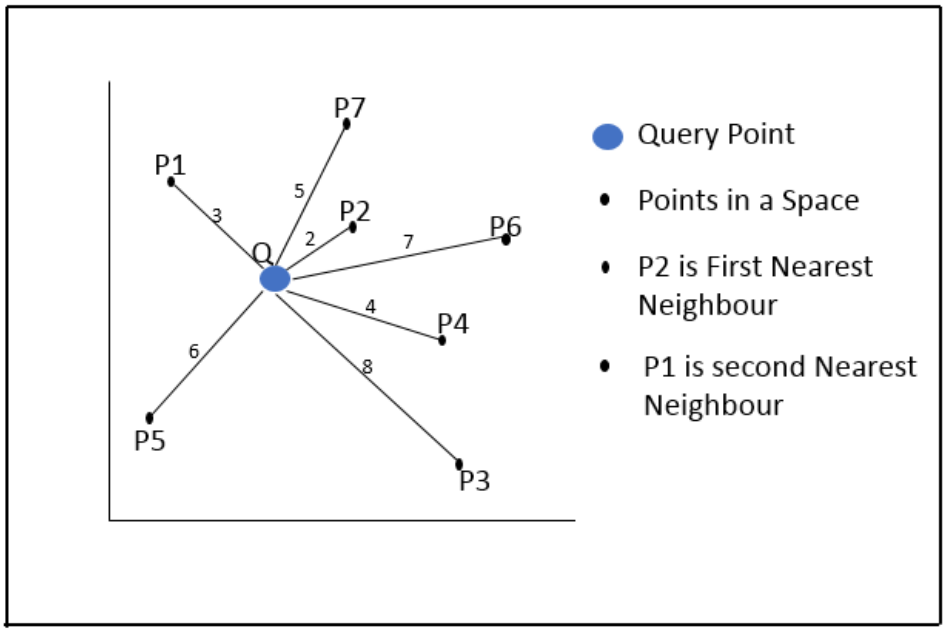

Figure 3.1: Nearest Neighbour example.

In Figure 3.1, we see several points P1 to P7 and the query point Q. Based on distances we see that P2 has the lowest distance to query point Q among all the points. So, it is termed as the first Nearest Neighbour (1NN). In addition, point P1 is termed as the second Nearest

Neighbour (2NN) of Q because it is the point which is the second closest point to Q. The remaining points are ordered similarly based on their distances to the query point.

3.2.1 Group Nearest Neighbour

Given a set of queries, Q= $\{q_1, ..., q_n\}$, and data points, P=$\{p_1, ..., p_m\}$, a Group Nearest Neighbour (GNN) query retrieves the data point from P that minimizes the total distance to all query points in Q.

In our work, we process the Group Nearest Neighbour query using the Multiple Query Method (MQM) which is based on the Threshold algorithm [8]. In MQM, an incremental Nearest Neighbour search is performed. We demonstrate MQM with the query set Q ={ C_{ens}, C_{end} }, which represent the centroids of the source and destination locations of a group of users.

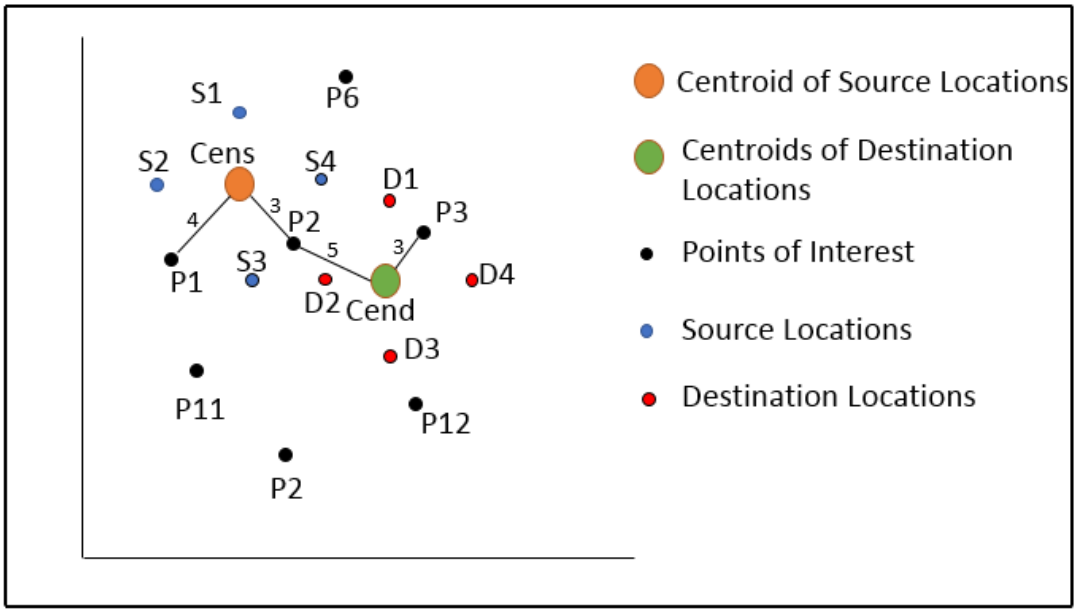

Figure 3.2: Group Nearest Neighbour example: GNN = P2.

Figure 3.2 shows an example of a GNN query. We can see that C_{ens}, C_{end} which are the two query points in Q=$\{C_{ens}, C_{end}\}$. MQM begins by finding an initial GNN. First, MQM finds the first NN of C_{ens} which is P2 (d(C_{ens}, P2) = 3). Then, the distance from P2 to the both query points in Q is calculated, (d(Q, P2) = d(C_{ens}, P2) + d(P2, C_{end}) = 3 + 5 = 8).

Next, MQM retrieves the first NN of C_{end} which is P3 (d(C_{end},P3) = 3) and then calculates the distance from P3 to the both query points in Q, (d(Q, P3) = d(C_{ens}, P3) + d(P3, C_{end}) = 6 + 3 = 9). P2 has the minimum sum of distances to all the query points in Q and becomes the current GNN with the best distance found so far. MQM then uses the threshold algorithm to test for other possible points that may generate a shorter distance than the current distance.

MQM calculates a condition $T \leq$ d(bestdistance) to find any possible point which has a shorter distance to all query points in Q than the point with the current best distance (i.e. P2). We calculate T by summing the distances from each query point to its current nearest neighbour. From Figure 3.2, the distance from C_{ens} to its first NN (P2) is called the current threshold of C_{ens}, t$Cen_s = 3$ and the distance from C_{end} to its current NN (P3) is called the current threshold of C_{end}, $tC_{end} = 3$. The current total threshold is $T = tC_{ens} + tC_{end} = 3 + 3 = 6$. Since T$\leq$d(bestdistance) (i.e. $6 <= 8$), we find the next NN of C_{ens} which is P1 and we compute its distance to both query points in Q (d(Q, P1) = 9). The current total threshold will be updated as follows $T = tC_{ens} + tC_{end} = 4 + 3 = 7$. Since T $<$ (bestdistance) (i.e. $7 <= 8$), we compute the next NN of C_{end} which is P2 and we compute the distance between P2 and both query points in Q (d(Q, P2) = 8). The current threshold is updated as $T = tC_{ens} + tC_{end} = 4 + 5 = 9$. Since T$>$ d(bestdistance) (i.e. $9 => 8$), the algorithm terminates with P2 as the best GNN result found for both the query points in Q. According to Papadias. et al. [8], all remaining non-encountered points have a distance greater than T and therefore those points can not generate the optimal GNN.

3.3 California Road Network

Our work uses a modification of the original California dataset [1]. A road network contains information about nodes, edges and Points of Interest (POIs). In the road network, all nodes are connected by edges with one or more other nodes. Each node contains information about its Node ID and its co-ordinates. Each edge contains information about its Edge ID, Start Node ID, End Node ID and Edge Length between the start node and the end node.

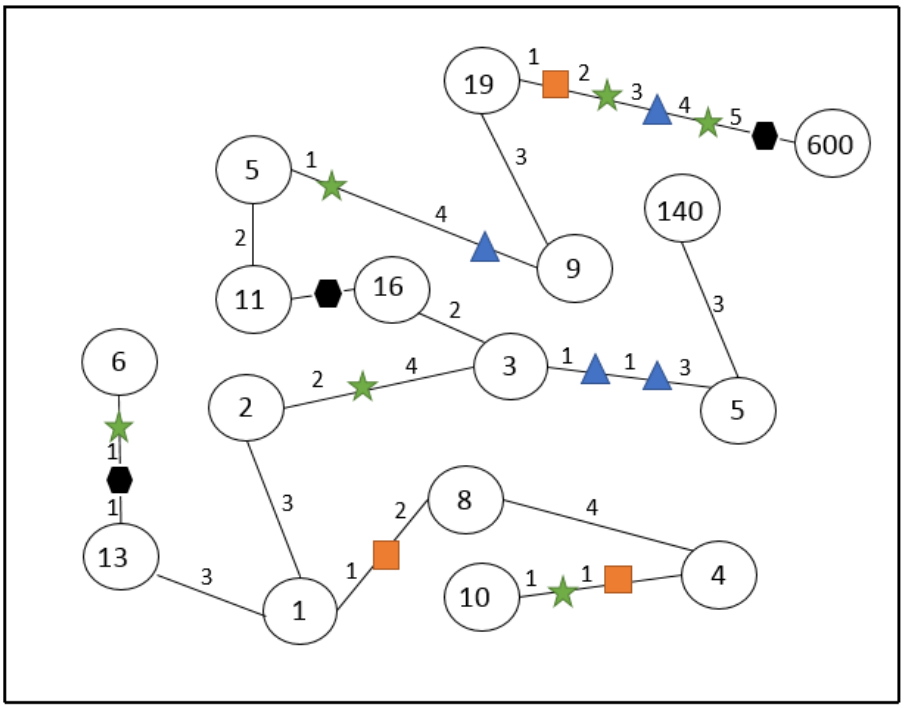

Figure 3.3: A sample road network.

A Category of Interest (COI) is a collection of Points of Interest (POIs) which provides the same service. An instance of a Category of Interest is then called a Point of Interest. For example, a bank is a Category of Interest, where instances of different banks such as Scotia Bank and RBC are called Points of Interest. The related map-format data contains information about nodes, edges, how many COIs lie on each edge and the distance(s) from the start node to each particular COI on its edge.

The following Figure 3.3 is a sample representative road network containing nodes, edges and POI information. The circles represent nodes with their associated Node ID. The straight lines represent edges that connect various nodes. The star, hexagon, square and triangle symbols that lie on the edges represent POIs. Each symbol represents an instance of a specific COI.

The edge length between two adjacent nodes is known as the network distance between these nodes. For example, from Figure 3.3, the network distance (i.e. spatial network

distance) from node 1 to node 2 is 3. The network distance between any two nodes is the sum of the network distances of the edges connecting them. For example, suppose we need to calculate the network distance between node 1 to node 5. The network distance is calculated by adding all edges lengths between node 1, 2, 3 and 5. The total network distance is 10. Similarly, we can calculate the network distance between a node and a POI. For example, to calculate the network distance from node 1 to the nearest star symbol COI which lies on the edge between nodes 2 and 3, first we calculate the network distance from node 1 to node 2 and then we add the distance to the POI, which is 2 to the total network distance. Therefore, the total network distance is 5.

We use Dijkstra's algorithm [6] in this thesis work to find the shortest path between a pair of nodes.

3.3.1 Data Generation

The current California road network data has a limitation. The related map-format data only states the COI IDs of the POIs that lie on that edge, but does not give the associated co-ordinates for the POIs. The pruning strategies that are used in the PGNE approach are based on the Euclidean distance, which requires co-ordinate information. Also, we use the POI co-ordinate information in the processing of the Group Nearest Neighbour queries.

To handle this limitation in the existing data, Ahmadi and Nascimento [2] processed Sequenced Group Trip Planning Queries (SGTPQs) with their PGNE algorithm by only using a small subset of the COIs in the California dataset. Although this will produce accurate results, their results are only shown for these chosen COIs. Although not specifically mentioned by the authors, we assume they encountered the same difficulties with the California dataset. However, having an accurate representation of the entire real-world spatial network is required for accurate results.

Therefore, to process SGTPQs in a real-world scenario and to overcome the limitations of the existing data, we created a new dataset using the Nearest Neighbour strategy. The

new data not only supports all the existing COIs in the California road network, but also in their actual locations.

```
0 1 0.002025 0
0 6 0.005952 0
1 2 0.01435 2
25 0.00537134 55 0.0040002
2 3 0.012279 1
49 0.00474167
3 4 0.011099 3
33 0.000301746 48 0.00971851 54 0.00138718
5 6 0.006157 0
5 7 0.001408 0
5 8 0.012008 1
18 0.0105136
7 265 0.003213 1
44 0.00181596
```

Figure 3.4: Map Format data.

We first discuss the existing data in more detail before we explain the generation of the new data using the Nearest Neighbour strategy. Figure 3.4 presents a subset of data from the California data set. The first line of Figure 3.4 states that there is a start node with an ID of 0, an end node with an ID of 1, an edge length of 0.002025 and 0 which represents that there are no COI instances on this edge. However, for the third line, we see that there is a start node with an ID of 1, an end node with an ID of 2, an edge length of 0.01435 and the number 2, which represents that there are two COI instances along this edge. One is a COI ID of 25, with a distance 0.005371 from the start node. The other COI ID is 55, with a distance of 0.00400 from the start node. From the data, we can clearly see that there is no other specific POI information such as co-ordinate information that we require to prune the search space when processing SGTPQs and AGTPQs. Rather, it only states which COI instance(s) lies on that edge.

Node-ID	Longitude	Latitude
0	-121.904	41.97456
1	-121.902	41.97477
2	-121.897	41.98808
3	-121.89	41.99803
4	-121.887	42.00874
5	-121.915	41.97031
6	-121.91	41.97394
7	-121.916	41.96948
8	-121.903	41.96846
9	-122.553	41.89524
10	-122.543	41.90308
11	-122.547	41.91573

Figure 3.5: Node co-ordinates.

Longitude	Latitudet	cat_id
-114.186	34.30806	0
-114.431	34.5275	1
-114.527	33.86944	21
-114.575	34.18389	29
-114.602	34.81944	11
-114.604	33.60361	29
-114.605	33.59194	70
-114.623	34.76611	2
-114.643	33.67917	40

Figure 3.6: POI coordinates with COI ID.

start_node	cat_id	distance	Longitude	Latitude
1	25	0.005371	-114.230003	34.29528
1	55	0.004	-114.254173	34.281109
2	49	0.004742	-114.78611	34.137779
3	33	0.000302	-114.471672	34.022221
3	48	0.009719	-114.138893	34.296391
3	54	0.001387	-114.174438	34.247219
5	18	0.010514	-114.657501	33.03194
7	44	0.001816	-114.14222	34.28722
9	40	0.000205	-114.405281	34.557781
9	43	0.00496	-114.141113	34.286388
9	44	0.007763	-114.14222	34.28722
9	50	0.004452	-114.144997	34.28833

Figure 3.7: A modified california dataset.

We used a Nearest Neighbour algorithm to overcome the limitations of the existing dataset. To do so, we use the following information. Figures 3.5 shows data about the node co-ordinates along with each node ID, while Figure 3.6 shows some POI coordinates along with the COI ID. We present a sample of newly formed data in Figure 3.7.

Referring back to Figure 3.4, we process the third line in the following manner. First, we take the co-ordinates for start node 1 and all the POI Co-ordinates for the 25th Category of Interest. Then we compute the Nearest Neighbour distances from each POI to start node 1. The POI which has the minimum distance to start node 1 is considered as the POI that lies on this edge. Likewise, we process the remaining COIs of the remaining data.

3.4 PGNE Approach and Running Example

We use the PGNE approach [2] to process the alternate sequenced group trip plans that are determined by our three strategies. PGNE creates a set of partial group trips (PGT) incrementally and all partial group trips are inserted into a priority queue based on the

lower bound of the group total travel distance. For example, each PGT can be in the forms of PGT(P1), PGT(P1, P2), PGT(P2), PGT(P3), PGT(P2, P4), etc., when they are inserted into the queue. The lower bound group travel distance of each PGT is defined as the sum of the length of the PGT plus the total Euclidean distances from the last POI in the PGT to the users' destinations. The authors defined the length of each PGT as the group's total spatial network distance from the users' starting locations to the last POI in the PGT. Each PGT is extracted from the top of the priority queue and processed through replacement and progressive operations. In the replacement operation, the POI of the last COI in the PGT will be replaced with another POI, which is the next group nearest neighbour from the same COI. The progressive operation will append the first group nearest neighbour POI of the next remaining COI defined for the trip to the extracted PGT.

In addition, pruning strategies are used in finding the optimal path by not inserting the updated PGTs into the priority queue if it satisfies the pruning conditions. The authors proposed three pruning strategies to prune the PGTs that can not generate a shorter network distance than the current minimum distance. One pruning strategy is for the progressive operation and the other two are for the replacement operation. The PGTs which satisfies the following pruning conditions are pruned.

For a PGT created from a replacement operation, the authors first check the number of POIs in the updated PGT,

If the updated PGT contains only one POI then the following pruning strategy 1 is applied:

$d_e(P^1h, C_{ens}) + d_e(P^1h, C_{end}) > (MinDist + d_e(C_{ens} + d_e(C_{ens}, D))/n.$

Here P^1h is the next GNN of the corresponding COI, MinDist is the minimum distance found so far, C_{ens} is the centroid of the source locations, C_{end} is the centroid of the destination locations and n is the number of users.

If the PGT contains more than one POI then the following pruning strategy 2 is applied:
$d_e(Ph-1, P^1h) + d_e(P^1h, C_{end}) > (MinDist - len(PGT) + d_e(C_{ens}, D))/n$

Here P^1h is the next GNN of the corresponding COI, Ph-1 is the POI of the previous COI contained in the PGT, MinDist is the minimum distance found so far, C_{ens} is the centroid of the source locations, C_{end} is the centroid of the destination locations and n is the number of users.

For a PGT created from a progressive operation the following pruning strategy 3 is applied:

$d_e(Ph, ph+1) + d_e(Ph+1, C_{end}) > (MinDist - len(PGT) + d_e(C_{ens}, D))/n$

Here Ph is the GNN of the corresponding COI, Ph+1 is the POI of the next COI in the visiting sequence, MinDist is the minimum distance found so far, C_{ens} is the centroid of the source locations, C_{end} is the centroid of the destination locations and n is the number of users.

3.4.1 Running Example of PGNE Strategy

We present the following example to illustrate the PGNE approach. A group of friends wants to visit a sequenced order of COIs while travelling from their start locations to their destinations. The Euclidean distances are assumed for the following example. Figure 3.8 shows the sample road networks used for this example. We assume the following trip information:

- Number of members in group, n = 4
- Set of source locations, S= {S1, S2, S3, S4} = {2, 8, 10, 42}
- Set of destination locations, D={D1, D2, D3, D4} = {7, 11, 4, 170}
- Number of COIs, m=3
- Sequenced order of COIs, C={C1, C2, C3} = {Movie theater (P), Bank (R), Park (T)}

We have explained about the replacement and progressive operations in the Section 3.4 but for the sake of our example demonstration, we restate them here:

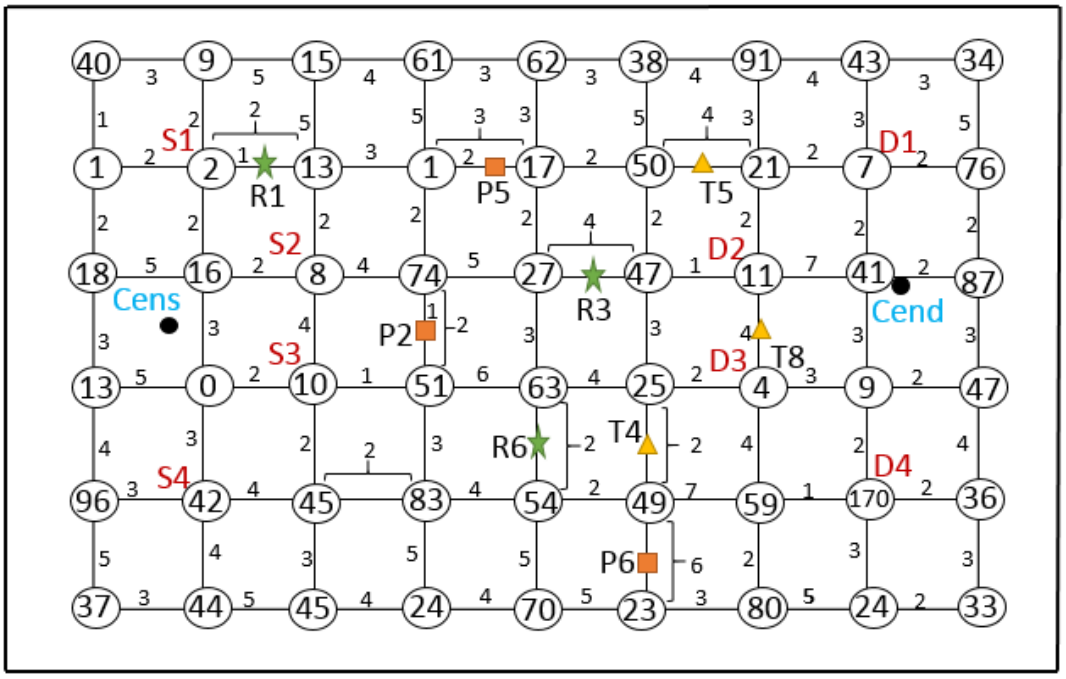

Figure 3.8: A Sample Road Network

- Replacement Operation: The POI of the last COI in the PGT will be replaced with another POI which is the next GNN from the same COI.

- Progressive Operation: Based on the COI order and the current PGT, the first GNN POI of the next remaining COI will be appended to the PGT.

First, we need to calculate C_{ens} and C_{end}. C_{ens} is S = (S1+S2+S3+S4)/4 = ((-121.89679, 41.988075) + (-121.903198, 41.968456) + (-122.542992, 41.903084) + (-121.902153, 41.974766))/4 = (-122.06128, 41.962621).

C_{end} is D = (D1+D2+D3+D4)/4 = ((-121.916199, 41.969482) + (-122.546921, 41.915726)+ (-121.886681, 42.008739) + (-122.92588, 41.912991))/4 = (-122.318920, 419517345).

Then, we need to calculate the first group nearest neighbour (GNN) query based on the C_{ens} and C_{end}. The GNN strategy is presented in the section 3.2.1. Using the Multiple Query Method (MQM), we find that P2 is the first GNN for the first COI C1 (Movie Theater).

Then, the first GNN, P2 is placed into a PGT as PGT(P2). Next, this PGT(P2) is inserted

into priority queue (PQ). During the process, all PGTs are stored in the priority queue in increasing order of their lower bound. PGT(P2) is the first to be inserted (and subsequently removed) from the priority queue.

For each iteration, we perform first a replacement, then a progressive operation, which produces two new PGTs. We also update the minimum distance if the current total distance of the PGT is less than the current minimum distance and PGT is equal to m. Finally we also check if the new PGTs satisfy the prunning strategies. The PGTs which do not satisfy the strategies will be pruned.

3.4.2 First Iteration

Initially there is only one PGT in PQ, which is PGT(P2). So we extract it from the queue. Then we perform replacement and progresssive operations on PGT(P2). We get PGT_1^1(P5) from the replacement operation. P5 is the second GNN for the movie theater COI from C_{ens} and C_{end}.

Since |PGT| $<$ m, we also perform a progressive operation on the PGT(P2) and we get PGT_2^1(P2, R3). R3 is the first GNN of the bank COI from P2 and C_{end}. Since $PGT_2^1 \neq$ m, we do not have to update the minimum distance and the final output. If a PGT contains all the POIs from the COI order, only then we do need to calculate the total distance of the PGT. Since the length of both PGT_1^1 and PGT_2^1 is not equal to m, we do not need to update the minimum distance. We need to apply pruning strategies on each PGT coming from those two operations.

The pruning strategy 1 is applied on PGT_1^1(P5):

$d_e(P5, C_{ens}) + d_e(P5, C_{end}) > (MinDist + d_e(C_{ens}, S) + d_e(C_{end}, D))/n = 15 + 20 > (\infty + 20 + 20)/n = 35 > \infty$.

Since the above pruning condition is false, we need to put PGT_1^1(P5) into PQ based on its lower bound.

The length of the PGT(P2) is $d_n(S1, P2) + d_n(S2, P2) + d_n(S3, P2) + d_n(S4, P2) = 8 +$

$5+2+7=21$.

The pruning strategy 3 is applied on PGT_2^1(P2, R3):

$d_e(P2,R3)+d_e(R3,C_{end}) > (MinDist - len(PGT)+d_e(C_{end},D))/n = 8+15 < (\infty - 21+20)/n = 23 > \infty$.

Since the above pruning condition is false, we need to put PGT_2^1(P2, R3) into PQ based on its lower bound.

We need to calculate the lower bound of each PGT in order to decide their position in the queue. Lower Bound=(Sum of the length of the PGT)+(Euclidean distances from the last POI in the PGT to the users' destination locations).

The Lower Bound Calculation of PGT_1^1 is $(d_n(S1,P5)+d_n(S2,P5)+d_n(S3,P5)+d_n(S4,P5))+(d_e(P5,D1)+d_e(P5,D2)+d_e(P5,D3)+d_e(P5,D4)) = (7+7+7+15)+(15+10+15+20) = 96$.

The Lower Bound Calculation of PGT_2^1 is $(d_n(S1,P2)+d_n(S2,P2)+d_n(S3,P2)+d_n(S4,P2))+(d_e(R3,D1)+d_e(R3,D2)+d_e(R3,D3)+d_e(R3,D4)) = (8+5+2+7)+(10+5+5+15) = 57$.

Based on the above lower-bound calculations, first PGT_1^1 and then PGT_2^1 are inserted. The priority queue is, PQ=$\{PGT_1^1, PGT_2^1\}$. As PQ is not empty, We repeat the same process.

3.4.3 Second Iteration

We extract PGT_1^1(P5) from the top of PQ, and perform replacement and progressive operations on it. We create PGT_1^2(P6) from the replacement operation and PGT_2^2(P5, R3) from the progressive operation. P6 is the next GNN of the movie theatre from C_{ens} and C_{end} and R3 is the first GNN of the bank from P5 and C_{end}.

The pruning strategy 1 is applied on PGT_1^2(P6):

$d_e(P6,C_{ens})+d_e(P6,C_{end}) > (MinDist+d_e(C_{ens},S)+d_e(C_{end},D))/n = 20+25 > (\infty+20+20)/n = 45 > \infty$.

Since the above pruning condition is false, we need to put PGT_1^2(P6) into PQ based on its lower bound.

The length of the PGT_1^1(P5) is $d_n(S1,P5)+d_n(S2,P5)+d_n(S3,P5)+d_n(S4,P5)=7+7+7+15=36$.

The pruning strategy 3 applied on PGT_2^2(P5, R3):

$d_e(P5,R3)+d_e(R3,C_{end}) > (MinDist - len(PGT_1^1)+d_e(C_{end},D))/n = 5+12 > \infty - 36+20 = 17 > \infty$.

Since the above pruning condition is false, we need to put PGT_2^2(P5, R3) into PQ based on its lower bound.

Lower Bound Calculation of PGT_1^2 is $(d_n(S1,P6)+d_n(S2,P6)+d_n(S3,P6)+d_n(S4,P6))+(d_e(P6,D1)+d_e(P6,D2)+d_e(P6,D3)+d_e(P6,D4)) = (21+17+13+15)+(20+12+8+10) = 116$.

Lower Bound Calculation of PGT_2^2 is $(d_n(S1,P5)+d_n(S2,P5)+d_n(S3,P5)+d_n(S4,P5))+(d_e(R3,D1)+d_e(R3,D2)+d_e(R3,D3)+d_e(R3,D4)) = (7+7+7+15)+(10+5+5+15) = 71$.

PGT_1^2 are PGT_2^2 inserted into queue based on their lower bound calculations. Therefore, PQ =$\{PGT_1^2, PGT_2^2, PGT_2^1\}$.

3.4.4 Third Iteration

We extract PGT_1^2 from PQ and perform replacement and progressive operation on it. We get PGT_2^3=(P6, R6) from the progressive operation and we do not perform replacement operation on PGT_1^2 since there are no more POIs of the Movie Theatre in the Figure 3.2. R6 is the first GNN of the bank from P6 and C_{end}.

The pruning strategy 3 is applied on PGT_2^3(P6, R6):

$d_e(P6,R6)+d_e(R6,C_{end}) > (MinDist - len(PGT_1^2)+d_e(C_{end},D))/n = 2+15 > \infty - 66+20$.

Since the above pruning condition is false, we need to put PGT_2^3(P6, R6) in PQ based

on its lower bound.

Lower Bound Calculation of PGT_2^3 is $(d_n(S1,P6)+d_n(S2,P6)+d_n(S3,P6)+d_n(S4,P6))+(d_e(R6,D1)+d_e(R6,D2)+d_e(R6,D3)+d_e(R6,D4))=(21+17+13+15)+(20+12+7+15)=120$.

However, PGT_2^3 with its calculated lower bound is inserted into PQ. So the priority queue now contains, PQ= $\{PGT_2^3, PGT_2^2, PGT_2^1\}$

3.4.5 Fourth Iteration

PGT_2^3(P6, R6) is extracted and the two operations are performed on it. We obtain PGT_1^3(P6, R3) from the replacement operation and PGT_2^4(P6, R6, T4) from the progressive operation. R3 is the next GNN of the Movie Theatre COI from P6 and C_{end} and T4 is the first GNN of the Park COI from R6 and C_{end}.

The pruning strategy 2 is used for PGT_1^3(P6, R3):

$d_e(R6,R3)+d_e(R3,C_{end})>(MinDist-len(PGT_2^3)+d_e(C_{end},D))/n=5+12>(\infty-66+20)/n=15>\infty$.

Since the above pruning condition is false, we need to put PGT_2^4(P6, R3) into PQ based on its lower bound.

The pruning strategy 3 is used for PGT_2^4(P6, R6, T4):

$d_e(R6,T4)+d_e(T4,C_{end})>(MinDist-len(PGT_2^3)+d_e(C_{end},D))/n=5+12>(\infty-66+20)/4=17>\infty$.

Since the above pruning condition is false, we need to put PGT_2^4(P6, R6, T5) into PQ based on its lower bound.

Lower Bound Calculation of PGT_1^3(P6, R3) is $(d_n(S1,P6)+d_n(S2,P6)+d_n(S3,P6)+d_n(S4,P6)+(d_e(R3,D1)+d_e(R3,D2)+d_e(R3,D3)+d_e(R3,D4))=(21+17+13+15)+(10+5+5+15)=101$.

Lower Bound Calculation of PGT_2^4 (P6, R6, T4) is $(d_n(S1,P6)+d_n(S2,P6)+d_n(S3,P6)+d_n(S4,P6)+(d_n(P6,R6))n)+(d_e(T4,D1)+d_e(T4,D2)+d_e(T4,D3)+d_e(T4,D4))=(21+$

$17+13+15)+(6)4+(20+10+7+15)=142$.

Since | PGT_2^4 | = m, we calculate the total distance of PGT_2^4 from the users' source locations through the order of POIs, and to their destinations.

The total travel distance of PGT_2^4 is $d_n(S1,P6)+d_n(S2,P6)+d_n(S3,P6)+d_n(S4,P6)+(d_n(P6,R6)+d_n(R6,T4))n+d_n(T4,D1)+d_n(T4,D2)+d_n(T4,D3)+d_n(T4,D4)$ is 127.

The total travel distance is updated to minimum distance found so far.

In addition, the partial group trips PGT_1^3, PGT_2^4 are inserted into the queue based upon satisfying the prunning condition. So the queue contains, PQ={ PGT_2^4,PGT_1^3, PGT_2^2, PGT_1^2}.

3.4.6 Fifth Iteration

PGT_2^4 is extracted and we perform only replacement operation on it because it is equal to m. We obtain PGT_1^5 (P6, R6, T8) from the replacement operation. T8 is the next GNN of the park from R6 and C_{end}.

The pruning strategy 3 is applied on PGT_2^5(P6, R6, T8):

$d_e(R6,T8)+d_e(T8,C_{end})>(MinDist-len(PGT_2^4)+d_e(C_{end},D))/n=10+7>(127-90+20)/4=17>14.25$.

Since the above pruning condition is True, we do not need to put PGT_2^4(P6, R6, T8) into PQ.

Since | PGT_2^5 | = m, we calculate the total distance of PGT_2^5 from the users' source locations through the order of POIs, and to their destinations.

The total travel distance of PGT_2^5 is $d_n(S1,P6)+d_n(S2,P6)+d_n(S3,P6)+d_n(S4,P6)+(d_n(P6,R6)+d_n(R6,T8))n+d_n(T8,D1)+d_n(T8,D2)+d_n(T8,D3)+d_n(T8,D4)$ is $21+17+13+15+(6+9)4+4+2+2+7$ is 141.

Since the total travel distance of PGT_2^5 is not less than the MinDist. We do not need to update the minimum distance. So the queue contains, PQ={,PGT_1^3, PGT_2^2, PGT_1^2}.

3.4.7 Sixth Iteration

PGT_1^3(P6, R3) is extracted and the two operations are performed on it. We obtain PGT_1^6(P6, R1) from the replacement operation and PGT_2^5(P6, R3, T5) from the progressive operation. R1 is the next GNN of the movie theatre from P6 and C_{end} and T5 is the first GNN of the park from R3 and C_{end}.

The pruning strategy 2 is applied on PGT_1^6(P6, R1):

$d_e(R3,R1) + d_e(R1,C_{end}) > (MinDist - len(PGT_1^3) + d_e(C_{end},D))/n = (10+20) > (127-66+20)/4 = 30 > 20.25$.

Since the above pruning condition is True, we need not to put PGT_1^6(P6, R1) into PQ.

The pruning strategy 3 is used for PGT_2^5(P6, R3, T5):

$d_e(R3,T5) + d_e(T5,C_{end}) > (MinDist - len(PGT_1^3) + d_e(C_{end},D))/n = (10+11) > (127-66+20)/4 = 21 > 20.25$.

Since the above pruning condition is True, we need not to put PGT_2^5(P6, R3, T5) into PQ.

The minimum distance of $PGT_2^5 is (d_n(S1,P6) + d_n(S2,P6) + d_n(S3,P6) + d_n(S4,P6) + (d_n(P6,R3) + d_n(R3,T5))n + d_n(T5,D1) + d_n(T5,D2) + d_n(T5,D3) + d_n(T5,D4))$ is $21 + 17 + 13 + 15 + 56 + 4 + 4 + 8 + 5 = 141$.

Since TD>MinDist, the minimum distance is not updated. Therefore, PQ=$\{PGT_2^2$, $PGT_1^2\}$.

3.4.8 Seventh Iteration

PGT_2^2(P2, R3) is extracted and the two operations are performed on it. We obtain PGT_1^7(P2, R6) from the replacement operation and PGT_2^6(P2, R3, T8) from the progressive operation.

The pruning strategy 2 is used for PGT_1^7(P2, R6):

$d_e(R3,R6) + d_e(R6,C_{end}) > (MinDist - len(PGT_2^2) + d_e(C_{end},D))/n = (5+15) > (127-22+20)/n = 20 > 31.25$.

Since the above pruning condition is false, we need to put the PGT_1^7(P2, R6) into PQ based on its lower bound.

The pruning condition 3 is used for PGT_2^6 (P2, R3, T8):

$d_e(R3,T8)+d_e(T8,C_{end}) > (MinDist-len(PGT_2^2)+d_e(C_{end},D))/n = (5+5) > (127-22+20)/4 = 10 > 31.25$.

Since the above pruning condition is false, we need to put PGT_2^6(P2, R3, T8) into PQ based on its lower bound.

The total travel distance of PGT_2^6 is $(d_n(S1,P2)+d_n(S2,P2)+d_n(S3,P2)+d_n(S4,P2)+(d_n(P2,R3)+d_n(R3,T8))n+d_n(T8,D1)+d_n(T8,D2)+d_n(T8,D3)+d_n(T8,D4))$ is $8+5+2+7+48+6+2+2+5 = 85$.

Since TD<MinDist, the minimum distance is updated.

Lower Bound Calculation of PGT_1^7(P2 R6) is $(d_n(S1,P2)+d_n(S2,P2)+d_n(S3,P2)+d_n(S4,P2)+(d_e(R6,D1)+d_e(R6,D2)+d_e(R6,D3)+d_e(R6,D4)) = (8+5+2+7)+(15+10+5+10) = 62$.

Lower Bound Calculation of PGT_2^6 (P2, R3, T5) is $(d_n(S1,P2)+d_n(S2,P2)+d_n(S3,P2)+d_n(S4,P2)+(d_n(P2,R3))n+(d_e(T5,D1)+d_e(T5,D2)+d_e(T4,D3)+d_e(T4,D4)) = (21+17+13+15)+(6)4+(20+10+7+15) = 142$.

PGT_1^7 are PGT_2^6 inserted into queue based on their lower bound calculations. PQ = $\{PGT_2^2, PGT7_1, PGT_2^1\}$.

3.4.9 Eight Iteration

we extract PGT_2^2 is extracted and the two operations are performed on it. We obtain PGT_1^8(P5, R6) from the replacement operation and PGT_2^7(P5, R3, T4) from the progressive operation.

The pruning strategy 2 is used for PGT_1^8(P5, R6):

$d_e(R3,R6)+d_e(R6,C_{end}) > (MinDist-len(PGT_2^2)+d_e(C_{end},D))/n = (5+15) > (85-36+20)/n = 20 > 17.25$.

Since the above pruning condition is True, we need not to put PGT_1^7(P2, R6) into PQ.

The pruning strategy 3 is used for PGT_2^8(P5, R3, T4):

$d_e(R3,T4)+d_e(T4,C_{end}) > (MinDist - len(PGT_2^2)+d_e(C_{end},D))/n = (7+11) > (85-36+20) = 18 > 17.25$.

Since the above pruning condition is True, we need not to put PGT_1^8(P2, R3,T4) in to PQ.

The minimum distance of $PGT_2^8 is(d_n(S1,P5)+d_n(S2,P5)+d_n(S3,P5)+d_n(S4,P5)+(d_n(P5,R3)+d_n(R3,T4))n+d_n(T4,D1)+d_n(T4,D2)+d_n(T4,D3)+d_n(T4,D4))$ is $(7+7+7+15)+(11)4+(9+5+3+8)$=105.

Since TD>mindist, the minimum distance is not updated.

PQ = { $PGT7_1$, PGT_2^1}.

3.4.10 Ninth Iteration

We extract PGT_1^7 from the priority queue and perform two operations on it. We get PGT_1^9(P2, R1) from replacement operation and PGT_2^8(P2, R6, T5) from the progressive operation.

The pruning strategy 2 is used for PGT_1^9(P2, R1):

$d_e(R6,R1) + d_e(R1,C_{end}) > (MinDist - len(PGT_2^7) + d_e(C_{end},D))/n = (10+20) > (85-22+20) = 30 > 20.75$.

Since the above pruning condition is True, we need not to put PGT_1^9(P6, R1) in to PQ. The pruning strategy 3 is used for PGT_2^8= (P2, R6, T5):

$d_e(R6,T5)+d_e(T5,C_{end}) > (MinDist - len(PGT_1^7)+d_e(C_{end},D))/n = (5+5) > (85-66+20)/4 = 10 > 9.75$. Since the above pruning condition is True, we need not to put PGT_1^6(P6, R6, T5) in to PQ.

The minimum distance of $PGT_2^8 is(d_n(S1,P2)+d_n(S2,P2)+d_n(S3,P2)+d_n(S4,P2)+(d_n(P2,R6)+d_n(R6,T5))n+d_n(T5,D1)+d_n(T5,D2)+d_n(T5,D3)+d_n(T5,D4))$ is $(7+7+7+15)+(8+10)4+4+8+13 = 127$.

Since TD>MinDist, the minimum distance is not updated.

3.4.11 Tenth Iteration

we extract PGT_2^1 from the PQ and we cannot perform both operations on it. Since there are no more POIs of Bank and Park.

Finally, the algorithm terminates with an PGT(P2. R3, T8) as an optimal answer and the minimum distance is 85.

3.5 Alternative Group Trip Planning Queries and Examples

In this section, we propose three strategies, all of which have the goal to find a more cost-effective trip than the trip initially submitted by the users. The three proposed strategies - Permutation, Greedy and Random - will attempt to find a shorter path by considering different orderings of the COIs. We discuss the proposed approaches in detail below. All the approaches will use modified California dataset which includes the POI co-ordinates, and the PGNE approach for processing the alternate trip plans.

3.5.1 Permutation Strategy

The Permutation Strategy is the first strategy. We implemented the Permutation Strategy to use alongside the PGNE approach. We use this strategy to find the different orderings of the COI set. Given m, the number of COIs in the COI set, the number of possible permutations of COIs is: $P(m,k) = m!/(m-k)!$. However, since k = m that makes the number of permutations as: $P(m,m) = m!$.

For example, suppose there are three COIs, C ={Movie Theatre, Bank, Hospital}. We find six permutations from the given example. The permutations are PMCOI = {Movie Theatre, Bank, Hospital}, {Movie Theatre, Hospital, Bank}, {Bank, Movie Theatre, Hospital}, {Bank, Hospital, Movie Theatre}, {Hospital, Movie Theatre, Bank}, and {Hospital, Bank, Movie Theatre}.

We then apply the PGNE approach to each generated permutation and compare the distances. The permutation with the lowest minimum distance(MinDist) among all the permutations is chosen as the best AGTPQ.

Algorithm 1: Permutation Strategy

```
1  Input: S = {S1, S2,..., S_n}
2         D = {D1, D2,..., D_n}
3         COI = {C1, C2,.., C_n}
4  MinDist=∞
5  FinalResult={}
6  PMCOI=FindPermutations(COI)
7  for EACH P in PMCOI: do
8      TripDist=PGNE(P, S ,D)
9      if TripDist < Mindist then
10         Update(TripDist, Mindist, FinalResult, P)
11
```

In Algorithm 1: Permutation Strategy, Line 1-3 contains information about the users' Source S, Destination D locations and COIs that they plan to visit. In Lines 4-5, initially, FinalResult is set to Null and the minimum distance (MinDist) is set to Infinity. In Line 6, all the permutations for PMCOI are calculated.

In Lines 7- 11, for each permutation from PMCOI its trip distance is computed using the PGNE approach. Then the algorithm compares the computed trip distance with the current minimum distance. If it is found to be shorter, than the current minimum distance, then the algorithm updates the minimum distance and FinalResult. The algorithm terminates when the PMCOI list is empty.

3.5.2 Greedy Strategy

The Greedy strategy is our second strategy. This strategy finds one alternate order of COIs from the original sequence order. The alternate order of COIs is based on the greedy distance (denoted as d_G), which is described in detail below:

Given Q=(C_{ens}, C_{end}), where C_{ens} is the centroid of the source locations and C_{end} is the centroid of the destination locations, d_G is the sum of the distances from both C_{ens} and C_{end} to the first GNN of COI Ci:

$$d_G(Ci) = d_e(C_{ens}, GNN(C_i)) + d_e(C_{end}, GNN(C_i))$$

We used the Euclidean distance metric to calculate the Greedy distance because C_{ens} and C_{end} usually reside in the empty space of the road network and not on the edges. The Greedy strategy takes each COI from the original trip sequence and finds the distance from its first group nearest neighbour POI to the centroids of the users' source and destination locations. Once these distances have been calculated, the Greedy strategy orders the COIs in ascending order of their greedy distances. Finally, we run the PGNE approach using the newly generated COI order.

In Algorithm 2: Greedy Strategy, Line 1 -3 contains information about the users' Source S, Destination D locations and COIs that they plan to visit. Line 4 contains information about the centroids of S and D, respectively. In Lines 5-7, the DistList (DistList is the list that contains each COIs Greedy distance) is set to Null, the minimum distance (MinDist) is set to infinity and Order is set to Null. In Lines 8-9, the algorithm calculates $d_G(C_i)$ for each COI C_i. In Line 10-13, the algorithm searches for each COI with the next smallest Greedy distance, and then appends to Order. It terminates once it appends all the COIs to Order. In-Line 14, the algorithm computes the minimum distance using PGNE.

Algorithm 2: Greedy Strategy

```
Input: S = {S_1, S_2, ..., S_n}
       D = {D_1, D_2, ..., D_n}
       COI = {C1, C2,.., C_n}
       Q = {C_ens, C_end}
DistList = {}
MinDist = ∞
Order = {}
for each Ci in COI: do
    DistList.append(C_i, d_G(C_i))
while DistList is not empty do
    nextC_i = Smallest(DistList)
    Order.append(nextC_i)
    DistList = DistList - (nextC_i, d_G(nextC_i));
MinDist=PGNE(Order, S, D)
```

3.5.3 Running Example of Greedy Strategy

We use the example from Figure 3.8 to demonstrate the Greedy strategy. The initial sequence order of COIs C={C1, C2, C3} = {Movie theater (P), Bank (R), Park (T)}. Order is set to null.

The Greedy approach takes the first COI C1 (Movie Theatre) and calculates the Greedy distance from the group nearest theatre (P2) to the C_{ens} and C_{end} which is $d_G(C1) = d_e(C_{ens}, P2) + d_e(C_{end}, P2) = (3+7) = 10$. Then it takes the second COI C2 (Bank) and calculates the Greedy distance from the group nearest bank (R3) to C_{ens} and C_{end}, which is $d_G(C2) = d_e(C_{ens}, R3) + d_e(C_{end}, R3) = (5+6) = 11$. Then it takes the third COI C3 (Park) and calculates the Greedy distance from the group nearest park (T4) to C_{ens} and C_{end}, which is $d_G(C3) = d_e(C_{ens}, T4) + d_e(C_{end}, T4) = (6+3) = 9$.

Then the Greedy strategy retrieves each COI in increasing order of Greedy distance and appends it to Order, until all the COIs are selected. Based on the calculations above, the final sequence order of COIs is C={Park, Movie Theatre, Bank}. Finally, the PGNE approach is applied to the new sequence order to find the total travel distance.

3.5.4 Random Strategy

The Random Strategy is the third proposed strategy. It obtains a new COI order by selecting COIs randomly from the original order with no repetition of COIs. The goal is to attempt to quickly find a more optimal trip than the original trip. Once a new trip is selected, we run the PGNE approach on it to determine the distance.

Algorithm 3: Random Strategy

```
1  Input: S = {S1, S2,..., S_n}
2         D = {D1, D2,..., D_n}
3         COI = {C1, C2,.., C_n}
4  MinDist=∞
5  Random={}
6  while COI is not empty do
7      Ci=SelectRandom(COI)
8      append(Ci, Random);
9      COI=COI-Ci
10 end
11 MinDist=PGNE(Random, S, D)
```

In Algorithm 3: Random Strategy, Line 1-3 contains information about the users' Source S, Destination D locations and COIs that they plan to visit. In Lines 4-5, the minimum distance (MinDist) is set to infinity and Random is set to Null. In Lines 6-9, the algorithm randomly selects a COI and appends it to Random. This process continues until the COI sequence is empty. In Line 11, the algorithm uses PGNE to compute the minimum distance based on the randomly generated COI sequence.

3.6 Summary

In this chapter, we presented our three proposed strategies. A running example for the PGNE approach was also presented in this chapter. The next chapter presents the performance evaluation of our proposed strategies.

Chapter 4

Experiments and Evaluations

In this chapter, we discuss the methodology and results from our experiments. We conducted several experiments by comparing all three proposed approaches with the PGNE approach [2]. The PGNE approach is a standard and the first approach to focus on SGTPQs so we choose the PGNE approach to compare against our proposed approaches. The modified version of the California road network dataset in (described in Section 3.3.1) is used in all experiments.

The rest of the chapter is organized as follows. Section 4.2 presents the information about the experimental setup and dataset used for our experiments. Section 4.3 presents the information about performance metrics. Sections 4.4 presents the experimental results and Section 4.5 presents a summary of this chapter.

4.1 Experimental Setup

For our experiments, we used the modified California road network data which we presented in Section 3.2.1. The original California dataset contains 62 categories and 103,882 POIs. Tables 4.1 and 4.2 illustrate the total information on the California road network data and the subset data containing the COIs we used for our experiments.

The PGNE and all three proposed approaches are implemented using the Python programming language. The set of experiments are carried out on an IntelR CoreTM2 Duo CPU E8757 with 2.66GHz CPU and 1835 MB RAM which is powered by the CentOs Linux 7 operating system. We evaluated and compared our proposed strategies against the

Table 4.1: California Road Network information.

Title	Information
Number of nodes	21,047
Number of edges	21,692
Number of COIs	62
Number of POIs	103,882

Table 4.2: COIs used in our experiments.

Categories	POIs
Airport	995
Hospital	835
School	11,173
Bend	486
Crater	356
Island	178
Church	7680
Bar	273
Gap	673
Harbor	169
Lake	462

PGNE approach using a set of experiments as follows:

- Varying the dataset size.
- Varying the group size.
- Varying the number of COIs.

We ran each experiment ten times and took an average of the results. We randomly selected the inputs for our experiments, such as the source and destination locations, and the COIs. This information was predetermined before we ran the tests. The following table 4.3 contains information about the parameters that we chose for each set of experiments. We set one parameter to a default value and another to varying according to the particular experiment. For the dataset size (i.e number of nodes), we carried out the experiments using 20% to 100% of the dataset, with the default percentage of the data being 40%. The first

20% of the data contains the first 4200 nodes of the dataset, its edges and POIs. Similarly, the first 40% of the data contains 8400 nodes, while the first 60% of the data contains 12600 nodes and 80% of the data contains 16800 nodes. Finally, 100% of the data contains 21,047 nodes. The dataset size is set to 40% as default because it contains almost most of the data to generate the results The number of COIs is set to 3 as default because it is faster to generate results of PGNE and proposed approaches (i.e. especially for Permutation strategy). The group size is set to 4 as default because it is an ideal group size used in many previous approaches in calculating the group trip distance.

Table 4.3: Parameters used in our experiments.

Parameter	Range	Default
Dataset size	20%, 40%, 60%, 80%, 100%	40%
Number of COIs	2, 3, 5, 7	3
Groupsize	2, 4, 8, 16, 32	4

4.2 Performance Metrics

All proposed strategies are compared with the PGNE approach using the following performance metrics:

- **Total Travel Distance:** Total Travel Distance is the sum of the distances of n users starting from their source locations S = $\{S_1,.., S_n\}$ through the selected sequence of m POIs from the given COI set P = $\{P_1,.., P_m\}$, and to their particular destinations D = $\{D_1,..., D_m\}$.

$$TD(P) = \sum_{i=1}^{n} d_n(S_i, P_1) + n\sum_{i=1}^{m-1} d_n(P_i, P_{i+1}) + \sum_{i=1}^{n} d_n(P_m, D_i)$$

where $d_n(u, v)$ is the spatial network distance between locations u and v.

- **Running time:** The running time is the time required to find and display the chosen group trip result. The running time is calculated in seconds.

4.3 Experimental Results

We evaluated our proposed approaches against the PGNE approach using the group total travel distance and running time. Our experiments included varying the database size in the data set, varying the number of COIs and varying the group size in order to evaluate the performance. In Section 4.3.1, we discuss the results for varying the database size in the data set. In Section 4.3.2, we discuss the results for varying the number of COIs. In section 4.3.3, we discuss the results for varying the group size.

4.3.1 Effect of the Dataset Size

In this set of experiments, we vary the percentage of points in the data set from 20% (4200 nodes), 40% (8400 nodes), 60% (12600 nodes), 80% (16800 nodes), up to 100% (21,047 nodes). Figures 4.1 and 4.2 illustrate the results of the approaches in terms of processing time and total travel distance.

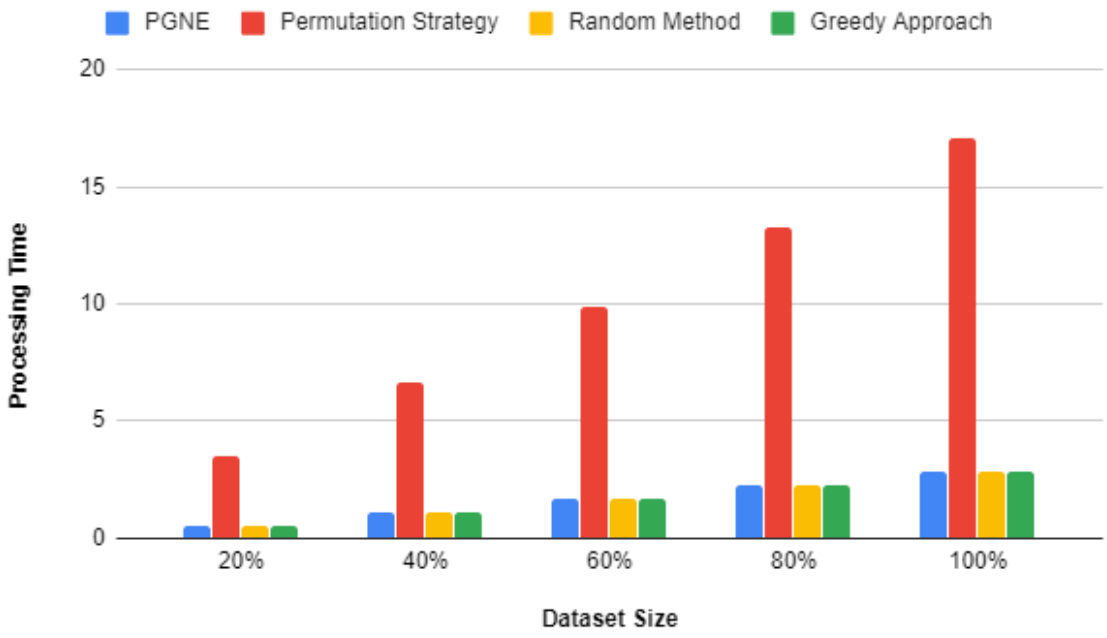

Figure 4.1: Effect of database size on processing time.

First, we discuss varying the dataset size against the processing times of all the techniques shown in Figure 4.1. We see that all the methods incur increasing processing time

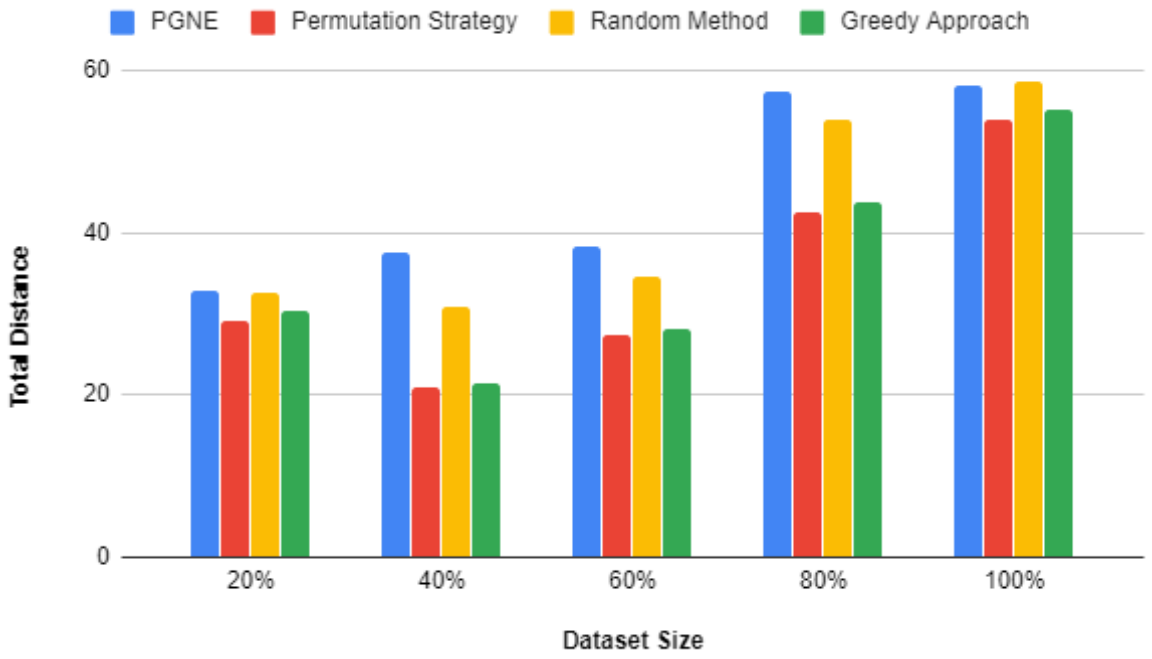

Figure 4.2: Effect of database size on total distance.

as the dataset size increases. This is because the increase in the dataset size leads to process more points. The experiment results show that the Greedy strategy, Random strategy and PGNE approach have almost the same processing time, where the Permutation strategy takes significantly more time to process the group trip result - approximately 300% more processing time than the other approaches. This is because the Permutation strategy processes the PGNE approach $P(m,k) = P(m,m) = m!$ number of times based on the number of permutations of m. The Greedy strategy and Random strategies has almost the same processing time as the PGNE approach because those approaches only run PGNE once. The PGNE approach outperforms the Permutation strategy in terms of generating group total travel distance in a shorter time. Therefore, the Greedy strategy, Random strategy and the PGNE approach are significantly better than the Permutation strategy in terms of processing time.

Second, we discuss varying the dataset size versus a group's total travel distance for all the strategies shown in Figure 4.2. We see that all the methods generate increasing total travel distances as the dataset size increases. However, the results also show that the

Permutation and Greedy strategies generate an average of a 15% lower total travel travel trip distance than the PGNE and Random strategies. The results in Figure 4.1 illustrate that the Permutation strategy takes significant time to process the AGTPQs, but the Greedy strategy generates the same trip distance as the Permutation strategy but with approximately 600% lower amount of time than the Permutation strategy. From Figure 4.2, both the Permutation strategy and Greedy strategy generate higher total trip distances for the first 20% of data than the first 40% of the data. This is because the newly generated COI sequence orders have lengthy distances between the randomly selected source and destination locations. All the proposed approaches generate equal or shorter group total travel distance than the PGNE approach. Overall, the Permutation and Greedy strategies outperforms the Random and PGNE approaches in all cases in terms of generating shorter distances.

4.3.2 Effect of Number of Categories

In this set of experiments, we use 40% data and four members of group to vary the number of COIs between 2, 3, 5 and 7. Figures 4.3 and 4.4 illustrate how much the approaches are affected by changing the number of COIs in the group trip.

In Figure 4.3, we see that all the methods incur increasing processing cost as the number of COIs increases. This is because of the increase in the number of COIs, which leads to processing a large number of NN query calculations for the GNN queries. The experiment results also show that the Greedy strategy, Random strategy and PGNE approach follow the same trend as the results for varying the data set size because they have almost the same processing time. Similarly, the Permutation strategy takes a significant amount of time to process the group trip result because of having to process $P(m,k) = P(m,m) = m!$ sets of COI orderings. The processing time for 5 COIs is 354 seconds while the processing time for 7 COIs is 26007 seconds. Therefore, the Greedy, Random and PGNE approaches are significantly better than the Permutation strategy in terms of processing time.

We now discuss varying the number of COIs versus the group's total travel distance

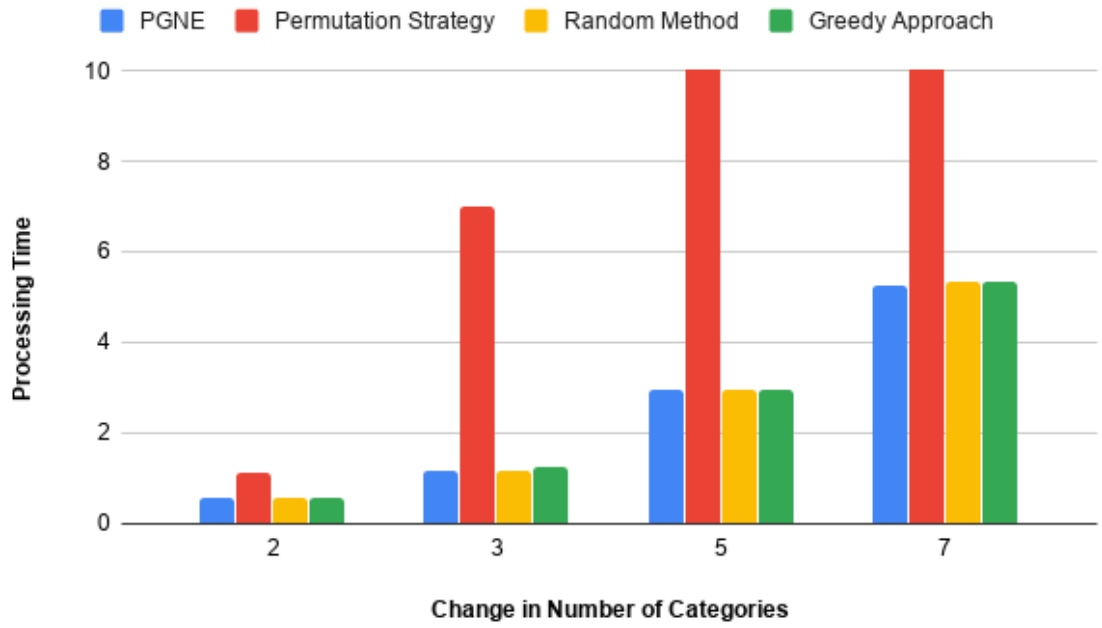

Figure 4.3: Effect of number of COIs on processing time.

for all the strategies shown in Figure 4.4. We see that all the strategies generate increasing total travel distances as the number of COIs increases. This is because the increase in the number of COIs will lead to additional distance being added onto each trip. We also notice that for a varying number of COIs different strategies follow different trends in generating total travel distances.

For 2 COIs, the PGNE approach generates a longer total trip distance than the Permutation, Greedy and Random strategies. However, this difference is not significant. For 3 COIs, the PGNE generates a 20% longer total trip distance than the Permutation and Greedy strategies. The Random and PGNE approaches generate the same total trip distances. For 5 COIs, the Permutation and Greedy strategies generate a 20% shorter trip distance than the PGNE approach. Also, the Permutation and Greedy strategies produce a 35% shorter trip distance than the Random strategy. For 7 COIs, the Permutation strategy generates a 40% shorter total travel trip distance than the PGNE, Random and Greedy approaches. From Figure 4.4, we see that the Greedy approach generates a shorter total travel distance from 2 to 5 COIs but the trip becomes longer for 7 COIs. This is due to processing more GNN

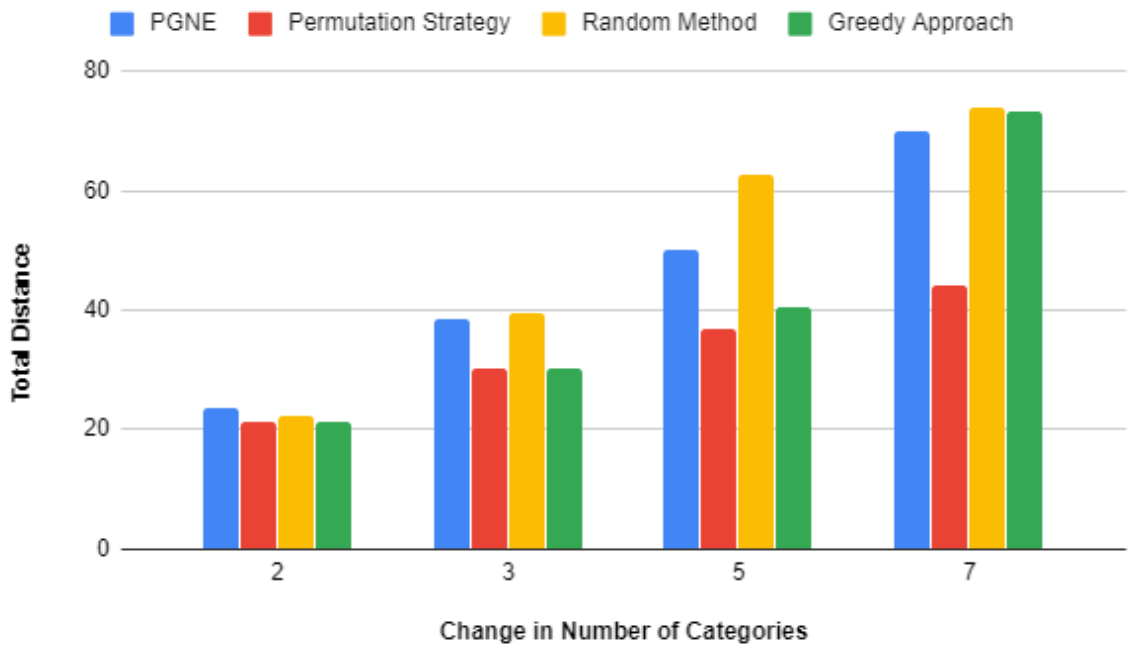

Figure 4.4: Effect of number of COIs on total distance.

calculations against the start and destination location centroids. Overall, the Permutation strategy generates an average 20% shorter trip than the PGNE approach.

4.3.3 Effect of Group Size

In this set of experiments, we compared the performance of our proposed approaches against the PGNE approach, when the group size increases between 2 and 32 users. Figures 4.5 and 4.6 presents the performance of all the approaches in terms of processing time and total travel distance.

From Figure 4.5, we observe first that all the approaches incur increasing processing cost as the number number of members in the group increases. This is because of the significant increase of the trip distance with the increase in the number of users in the group. The experiment results also show that the Greedy strategy, Random strategy and PGNE approach have almost the same processing time, while the Permutation strategy takes a significant amount of time to process the group trip result. Again, this is because many more alternative trips need to be processed.

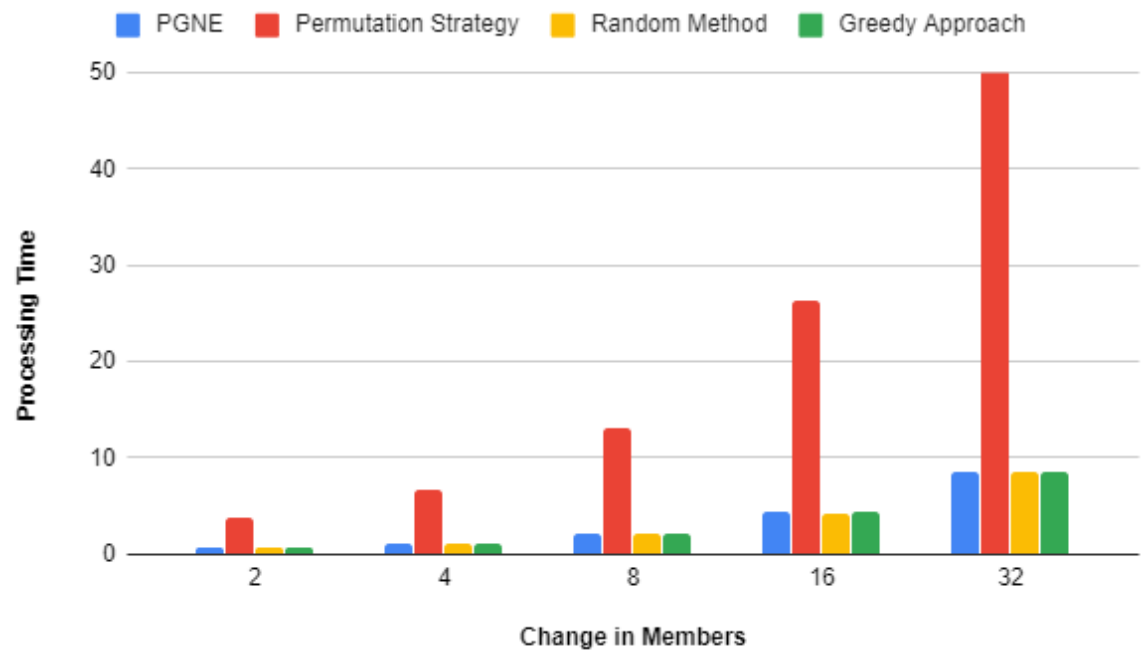

Figure 4.5: Effect of group size on processing time.

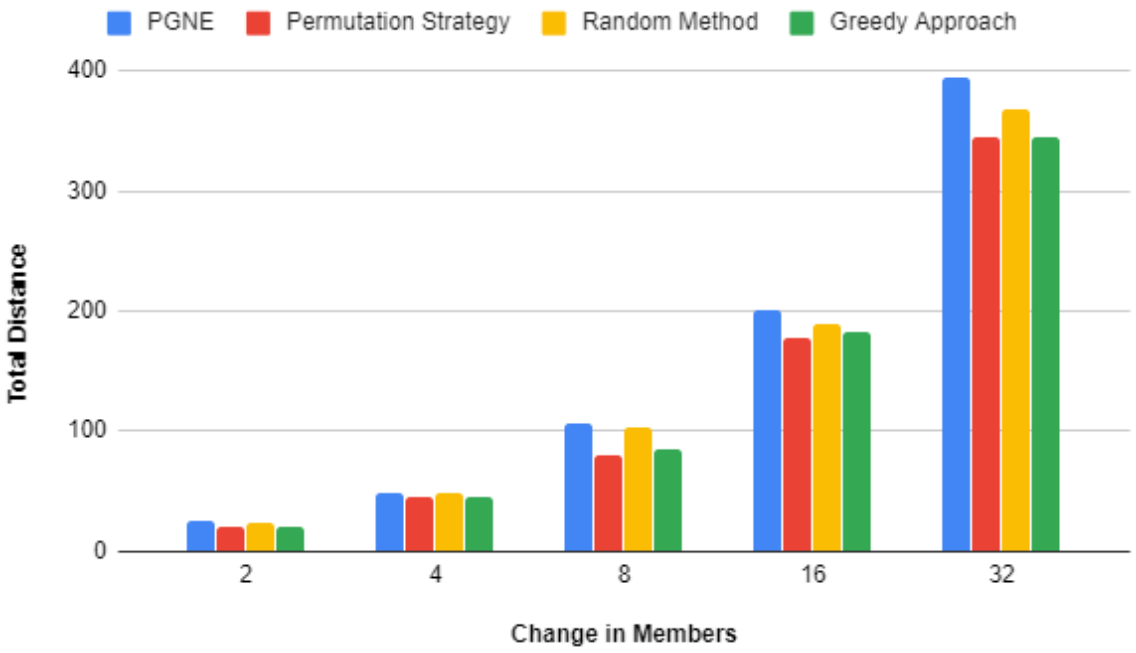

Figure 4.6: Effect of group size on total distance.

We now discuss varying the number of members in the group against the group's total distance of all the techniques shown in Figure 4.6. We see that all the strategies incur

increasing total travel distances as the group size increases. This is because of the increase in group size will lead to more distance calculation from the users' source and destination locations through the POIs. The Permutation and Greedy strategies follow the same trend as we have seen earlier. Both strategies produce shorter trips than the PGNE approach. In addition, the Random strategy generates a 7% shorter trip distance than the PGNE approach as the group size changes from 16 and 32.

4.4 Chapter Summary

In this Chapter, a set of experiments and comparisons were conducted for all the proposed approaches versus the PGNE approach. The results show that the Permutation strategy always finds an optimal trip but its processing time is high. From the results, the Greedy approach always finds a trip of equal or shorter distance than other approaches with a similar processing time to PGNE approach.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We propose three strategies to process the Alternative Group Trip Planning Queries (AGTPQs) in Location-Based Services (LBS). The alternative sequence order is generated by one of the proposed strategies with the goal to generate the shorter trip distance. In addition, the existing California dataset does not contain POI information and the existing approaches randomly added the POI information to the nodes which is not realistic in the real world. Therefore, we also generated a modified California dataset using the Nearest Neighbour strategy.

In our first strategy, we use the Permutation technique on the COI sequence to generate a number of different COI orderings based on the $P(m,k) = P(m,m) = m!$ formula. After finding the different permutations, we find the minimum total travel distance of each permutation using the PGNE approach. Then, we determine the minimum distance among all the permutations. In our second strategy, called Greedy, we use a distance function to reorder the predetermined sequence of the COI order. For each COI, it calculates the distance from centroids of the source and destination locations to the first GNN of that particular COI. Then, the COI order is sorted in increasing order based on the distance calculated for each COI. Then, we apply the PGNE approach to this new order to determine the minimum total travel distance. In our third strategy, called Random, we randomly select the COI order from the original COI sequence with no repetition. Then, we apply the PGNE approach to this new order to calculate the minimum total travel distance.

We evaluated our proposed strategies with various performance parameters. We found that the Permutation strategy produces a 30% lower trip distance than the remaining approaches but it takes a significant amount of time to process the AGTPQs. The Greedy strategy generates an equal or lower trip distance than the Random Strategy and PGNE approach. Also, the Random Strategy generates lower trip distances in some cases.

5.2 Future Work

In the future, we plan to retrieve the POIs based on the popularity generated from Google and Trip Advisor platforms to process the SGTPQs and AGTPQs.

In our Permutation Strategy, the same COIs are processed multiple times and this leads to a significant amount of processing time. So we plan to develop a technique which reduces the accessing of the same COIs multiple times and still produce an optimal trip.

We also plan to work on applying elliptical properties to refine the POI search space space, which are used by other strategies.

We also plan to work on processing Group Trip Planning Queries based on the user's affinity factor. Affinity factor [31] is the factor where the users in the group can decide with whom they want to travel that sub-sequence of the trip and meet in between the trip.

We also plan to work on finding an optimal point to start the group trip which produces the shortest path.

A Dynamic Group Trip Planning Query processes a Group Trip Planning Query in a way that any user in the group can leave or join at any POI. So, we also plan to improve AGTPQs where any user in the group can leave or join at any point of the trip.

We also plan to work on the Time-Dependent Group Trip Planning Query where we plan to take into account of other factors, such as time, traffic and public transportation, for processing AGTPQs.

Bibliography

[1] Spatial Network data. https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm. Accessed: 2019-02-20.

[2] Elham Ahmadi and Mario A. Nascimento. A mixed breadth-depth first search strategy for sequenced group trip planning queries. In *2015 16th IEEE International Conference on Mobile Data Management*, volume 1, pages 24–33, June 2015.

[3] Elham Ahmadi and Mario A. Nascimento. Ibs: An efficient stateful algorithm for optimal sequenced group trip planning queries. In *2017 18th IEEE International Conference on Mobile Data Management (MDM)*, pages 212–221, May 2017.

[4] Sukarna Barua, Roksana Jahan, and Toufique Ahmed. Weighted optimal sequenced group trip planning queries. In *2017 18th IEEE International Conference on Mobile Data Management (MDM)*, pages 222–227, May 2017.

[5] Haiquan Chen, Wei-Shinn Ku, Min-Te Sun, and Roger Zimmermann. The multi-rule partial sequenced route query. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, New York, NY, USA, 2008. Association for Computing Machinery.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[7] Camila F. Costa, Mario A. Nascimento, José A. F. Macundefineddo, Yannis Theodoridis, Nikos Pelekis, and Javam Machado. Optimal time-dependent sequenced route queries in road networks. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, New York, NY, USA, 2015. Association for Computing Machinery.

[8] Dimitris Papadias, Qiongmao Shen, Yufei Tao, and Kyriakos Mouratidis. Group nearest neighbor queries. In *Proceedings. 20th International Conference on Data Engineering*, pages 301–312, April 2004.

[9] Hicham G. Elmongui, Mohamed F. Mokbel, and Walid G. Aref. Continuous aggregate nearest neighbor queries. *GeoInformatica*, 17:63–95, 2011.

[10] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, page 47–57, New York, NY, USA, 1984. Association for Computing Machinery.

[11] Tanzima Hashem, Sukarna Barua, Mohammed Eunus Ali, Lars Kulik, and Egemen Tanin. Efficient computation of trips with friends and families. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 931–940, New York, NY, USA, 2015. Association for Computing Machinery.

[12] Tanzima Hashem, Tahrima Hashem, Mohammed Eunus Ali, and Lars Kulik. Group trip planning queries in spatial databases. In Mario A. Nascimento, Timos Sellis, Reynold Cheng, Jörg Sander, Yu Zheng, Hans-Peter Kriegel, Matthias Renz, and Christian Sengstock, editors, *Advances in Spatial and Temporal Databases*, pages 259–276, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[13] Mohammad Kolahdouzan and Cyrus Shahabi. Voronoi-based k nearest neighbor search for spatial network databases. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, page 840–851. VLDB Endowment, 2004.

[14] Chuanwen Li, Yu Gu, Jianzhong Qi, Ge Yu, Rui Zhang, and Wang Yi. Processing moving knn queries using influential neighbor sets. *Proc. VLDB Endow.*, 8(2):113–124, October 2014.

[15] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-Hua Teng. On trip planning queries in spatial databases. In *Proceedings of the 9th International Conference on Advances in Spatial and Temporal Databases*, SSTD'05, page 273–290, Berlin, Heidelberg, 2005. Springer-Verlag.

[16] Eric Hsueh-Chan Lu, Huan-Sheng Chen, and Vincent S. Tseng. An efficient framework for multirequest route planning in urban environments. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):869–879, April 2017.

[17] Kyriakos Mouratidis, Dimitris Papadias, and Marios Hadjieleftheriou. Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, page 634–645, New York, NY, USA, 2005. Association for Computing Machinery.

[18] Kyriakos Mouratidis, Man Lung Yiu, Dimitris Papadias, and Nikos Mamoulis. Continuous nearest neighbor monitoring in road networks. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, VLDB '06, page 43–54. VLDB Endowment, 2006.

[19] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. Query processing in spatial network databases. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, page 802–813. VLDB Endowment, 2003.

[20] Yeasir Rayhan, Tanzima Hashem, Roksana Jahan, and Muhammad Aamir Cheema. Efficient scheduling of generalized group trips in road networks. *ACM Trans. Spatial Algorithms Syst.*, 5(2), July 2019.

[21] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. Nearest neighbor queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, page 71–79, New York, NY, USA, 1995. Association for Computing Machinery.

[22] Maytham Safar. Group k -nearest neighbors queries in spatial network databases. *J Geogr Syst*, 10:407–416, 12 2008.

[23] Samiha Samrose, Tanzima Hashem, Sukarna Barua, Mohammed Eunus Ali, Mohammad Hafiz Uddin, and Md Iftekhar Mahmud. Efficient computation of group optimal sequenced routes in road networks. In *2015 16th IEEE International Conference on Mobile Data Management*, volume 1, pages 122–127, June 2015.

[24] Mehdi Sharifzadeh, Mohammad Kolahdouzan, and Cyrus Shahabi. The optimal sequenced route query. *The VLDB Journal*, 17(4):765–787, July 2008.

[25] Mehdi Sharifzadeh and Cyrus Shahabi. Processing optimal sequenced route queries using voronoi diagrams. *Geoinformatica*, 12(4):411–433, December 2008.

[26] Shashi Shekhar and Sanjay Chawla. Spatial databases - a tour. 2003.

[27] Nusrat Sultana, Tanzima Hashem, and Lars Kulik. Group nearest neighbor queries in the presence of obstacles. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '14, page 481–484, New York, NY, USA, 2014. Association for Computing Machinery.

[28] Anika Tabassum, Sukarna Barua, Tanzima Hashem, and Tasmin Chowdhury. Dynamic group trip planning queries in spatial databases. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, SSDBM '17, New York, NY, USA, 2017. Association for Computing Machinery.

[29] Yufei Tao and Dimitris Papadias. Time-parameterized queries in spatio-temporal databases. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD '02, page 334–345, New York, NY, USA, 2002. Association for Computing Machinery.

[30] Yufei Tao, Dimitris Papadias, and Qiongmao Shen. Continuous nearest neighbor search. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, page 287–298. VLDB Endowment, 2002.

[31] George Tsatsanifos, Alexandr Petcovici, and Mario A. Nascimento. Meet-and-go: Finding optimal single connecting points considering companionship preferences. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, New York, NY, USA, 2015. Association for Computing Machinery.

[32] Mala Veeresha and M. Sugumaran. Hybrid spatial air index for processing queries in road networks. *Cluster Computing*, 21:1–13, 03 2018.

[33] Xuedong Du and Jiangtao Ji. Research of knn query processing in road networks. In *2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS)*, volume 1, pages 72–76, Dec 2009.

[34] Man Lung Yiu, Nikos Mamoulis, and Dimitris Papadias. Aggregate nearest neighbor queries in road networks. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):820–833, June 2005.

[35] Geng Zhao, Kefeng Xuan, Wenny Rahayu, David Taniar, Maytham Safar, Marina L. Gavrilova, and Bala Srinivasan. Voronoi-based continuous k nearest neighbor search in mobile navigation. *IEEE Transactions on Industrial Electronics*, 58(6):2247–2257, 2011.